Scripts Reference Manual

Copyright © 1999 - 2015 Elipse Software Ltda. All rights reserved.
Version 4.7.234 (10/02/2015)

Table of Contents

1 Introduction 1
1.1 Objects 1
1.2 Scripts 2
1.3 Picks 7
1.4 User-Defined Events 14
2 Programming in E3 17
2.1 Programming Environment 17
2.2 Declaring Variables 18
2.3 Getting References to Objects 18
2.4 Accessing Objects 37
2.5 Working with Collections 38
2.6 Set Command 39
2.7 E3Globals 40
2.8 General Events, Methods, and Properties of Objects 47
3 User Libraries 69
3.1 XControls and XObjects 69
3.2 ElipseX Properties 71
4 View 74
4.1 Viewer 74
4.2 Frames and Splitters 101
4.3 Screens and Screen Objects 108
4.4 E3Alarm 223
4.5 E3Browser 246
4.6 E3Chart 254
4.7 E3Playback 297
4.8 Reports 298
5 Server Objects 338
5.1 Common Properties 338
5.2 Collection of Alarm's User Fields 340
5.3 Server's Runtime Objects 343
5.4 Configuration-Only Objects 346
5.5 Drivers 348
5.6 Data Server 405
5.7 Database 435
5.8 Historic 439
5.9 Storage 443

5.10 Formulas 450
5.11 Alarms 454
6 Frequently Asked Questions 481

CHAPTER 1

Introduction

Scripts are programming language modules that allow users to create procedures associated to specific events, providing them with higher flexibility when developing applications. Every E3 object (item of an application) has a list of previously defined events, but it is also possible to define new user-created events.

1.1 Objects

Objects are reusable software components that allow maximizing usage, and increasing quality and productivity of applications.

An E3 object encapsulates or contains three different parts (properties, methods, and events) that users can manipulate to obtain the advantages of its functionality in their application.

Properties define the object's attributes, such as the appearance of a Screen control, or an object's initial value when users start the application.

Methods are functions that perform a specific action with or within an object.

Events are notifications generated by an object as a response to some particular occurrence, such as a mouse click or a change in a Tag's value, among others.

One of the main features of objects and object-oriented languages is their ability of inheritance among each other, that is, they can inherit features of one or more objects, having the same specific functionality. So, users can have different objects working together to provide features of another derived object.

For example, take the object E3Chart. It consists internally of many objects, such as titles, legends, scales, divisions, queries, and pens. Notice that each object contributes for E3Chart's functionality as a whole: scales help locating point values, legends help identifying pens and their values, and pens help drawing values in the E3Chart.

By handling objects within the E3Chart, users can create two instances of this object that are very different from one another. To handle a specific object, users must access it through a hierarchy. If both E3Charts are on the same Screen, users must first access the Screen, then the preferred E3Chart, and then one of its properties or child objects. When there are many objects of the same type, usually they can be accessed through a collection. A collection is a special object that manages a set of similar objects. An example in the E3Chart is the Pens collection, which allows access to all E3Chart pens.

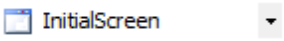
1.2 Scripts

The language that E3 Studio uses in its scripts is **VBScript**, a subset of the Visual Basic® language developed by Microsoft. VBScript has a fast, light, and portable interpreter, developed for use on Internet browsers and other applications using ActiveX Controls, Automation Servers, and Java Applets.

As seen before, scripts are linked to events of an object. However, to facilitate and increase development speed, E3 has already incorporated some of the most common actions that can be performed with scripts, through wizards known as **Picks**. Users can therefore define that a given event will either run a script, a Pick, or a combination of both, in a sequence that is also pre-defined.

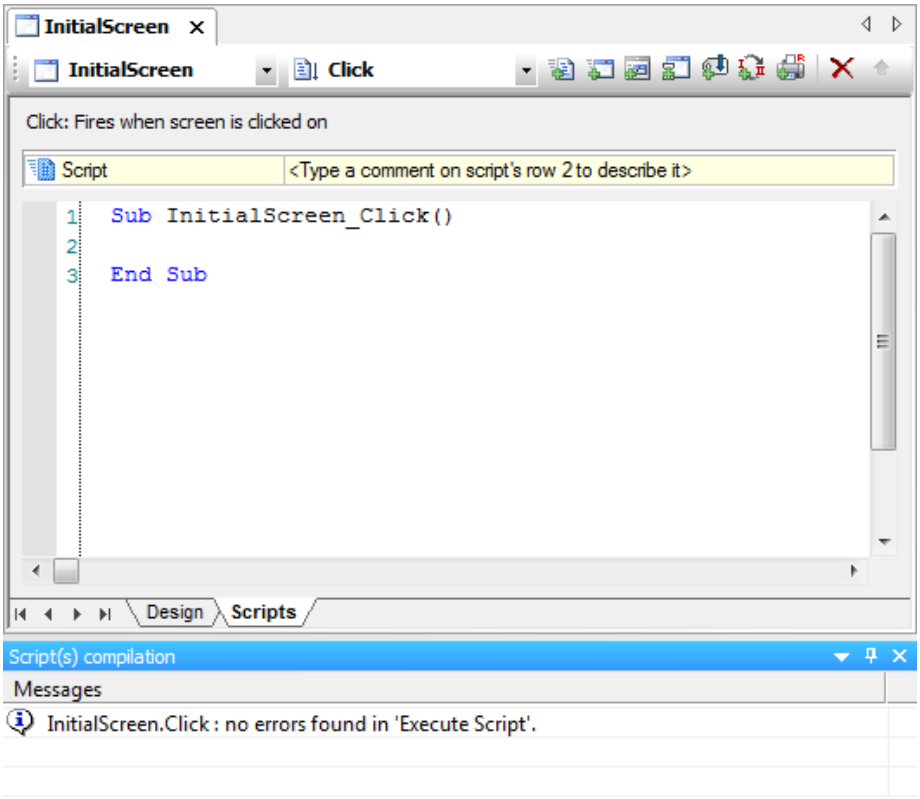
Each view in E3 Studio displays at least two tabs on its lower side: **Design** and **Scripts**, except for Database and Alarm Server objects, which do not have a **Design** tab. Objects and their child objects can be manipulated on the **Design** tab; to manipulate their scripts, use the **Scripts** tab. The available options on the latter are:

Available options for Scripts tab

ICON	OPTION	DESCRIPTION
	**List of Objects**	Selects the object whose script will be manipulated.
	List of Events	Selects the event to apply to an object.
	Script	Adds scripts linked to the event.
	Pick Open Screen	Adds an **Open Screen** Pick.
	Pick Open Modal Screen	Adds an **Open Modal Screen** Pick.
	Pick Run Application	Adds a **Run Application** Pick.
	Pick Load Value	Adds a **Load Value** Pick.
	Pick Toggle Value	Adds a **Toggle Value** Pick.
	Pick Print Report	Adds a **Print Report** Pick.
	Remove selected script or Pick	Removes the selected script or Pick from the Actions List.

ICON	OPTION	DESCRIPTION
	Move selected script or Pick up	Moves the selected action up, in the sort order of the Actions List for the event.
	Move selected script or Pick down	Moves the selected action down, in the sort order of the Actions List for the event.
	AppBrowser	Opens the AppBrowser window.
	Find	Finds occurrences of a given text.
	Find previous	Selects the previous text occurrence in the results list.
	Find next	Selects the next text occurrence in the results list.
	Replace	Replaces the occurrences in the results list by another specified text.
	Create user event	Creates a user-customized event.
	Remove user event	Removes the selected user-customized event.
	Edit user event	Edits the selected user-customized event.
	Compile selected script	Compiles the selected script, and shows its errors on the **Messages** panel.
	Compile all scripts from this event	Compiles all scripts linked to the event.
	Compile all events from this object	Compiles all events linked to the object.

The execution order of actions is top-down. To change this order, use the and options. Use the option to check for errors in the specified script for the event. The compiler's error messages are displayed on the **Messages** panel, which can appear as a floating window or docked on the lower or upper side of the **Scripts** tab. Double-click the error to select it in the script.

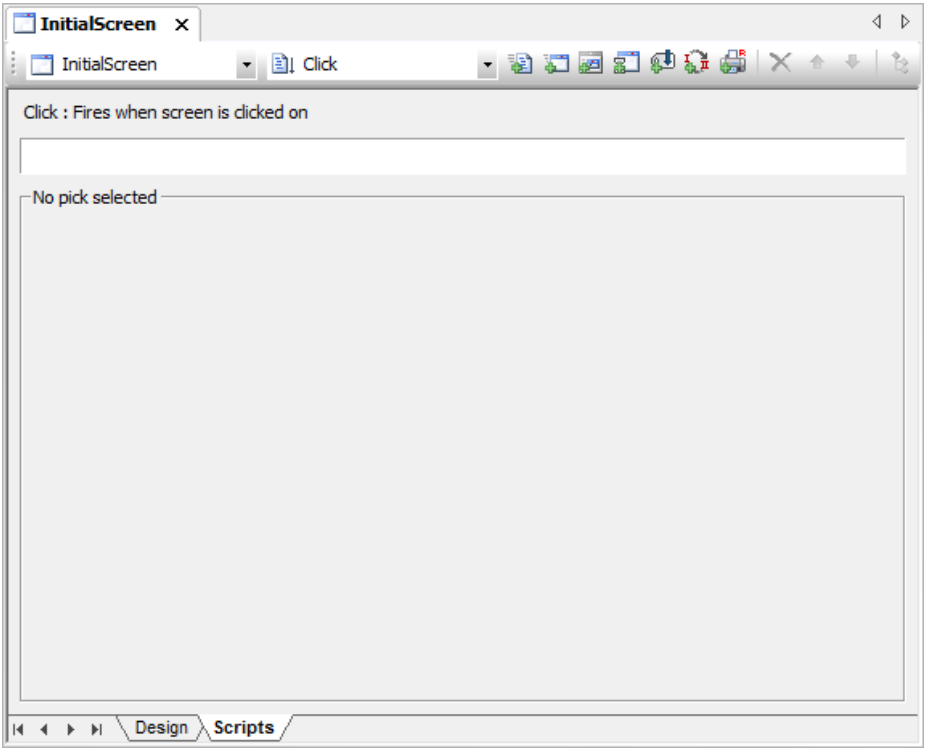


Compiler messages

1.2.1 Adding a Script

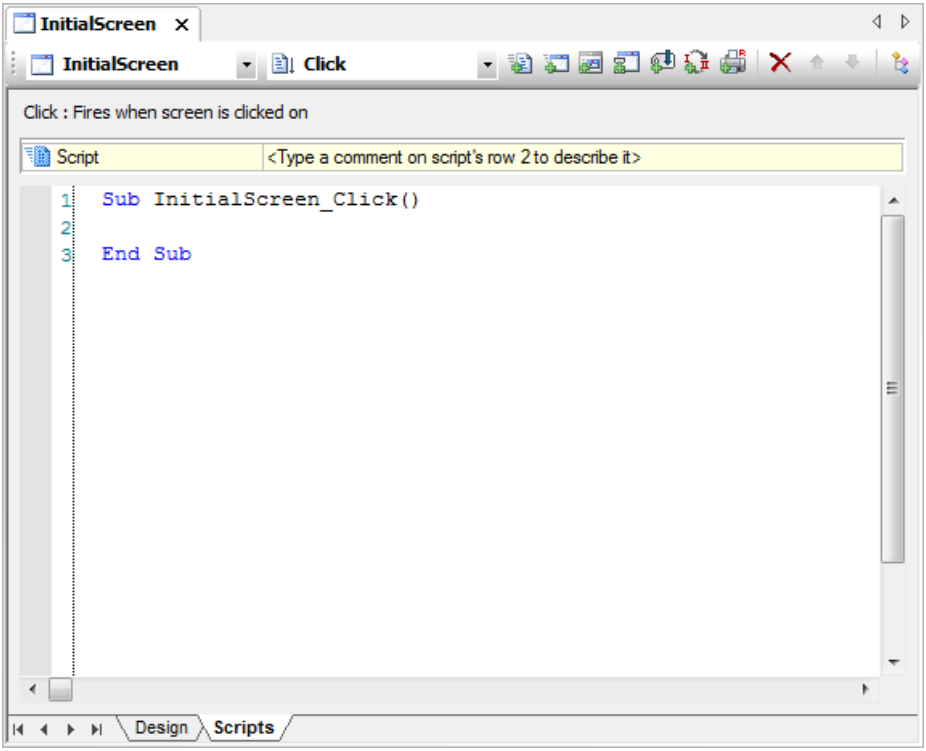
To add a script to an object, follow these steps:

1. Select the object to create the script, and then click the **Scripts** tab.



Scripts tab

2. Click the icon. The Scripts Editor is then opened, according to the next figure.



Adding a script to an object

3. Type the VBScript commands in the text-editing box.

NOTE: Use the underline character to add a line break, to make code more legible. The underline character indicates that the code continues on the next line.

For example:

```
If intBoilerTemperature3 > 120 and _
  intBoilerTemperature4 > 120 Then
  bSendAlarm = True
  bAlarmOn = True
End If
```

Each event can have several scripts and Picks linked to it, called **Event Actions**. The list of actions can be seen on the upper side of the scripts editing window. Each object can have any number of events with linked scripts or Picks.

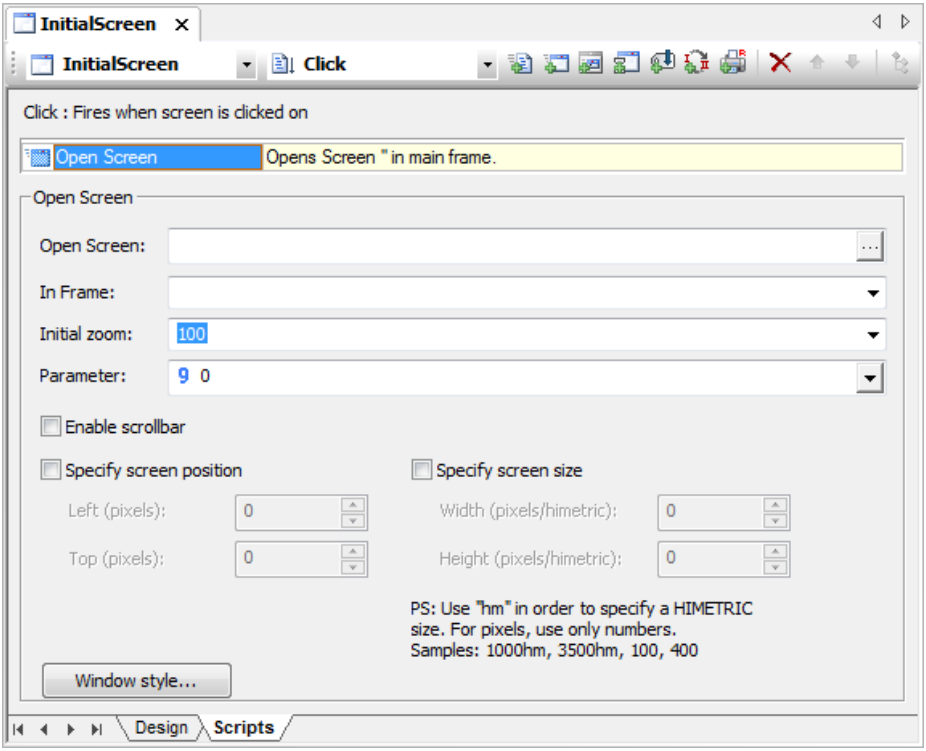
NOTE: When right-clicking any of these actions described previously, a contextual menu with options to cut, copy, and paste scripts and Picks among events is then opened.

1.3 Picks

Picks implement a friendlier way of performing the most common procedures, by saving configuration time. Among them, there are actions, such as switching Screens or attributing values, which are very common while creating a project. Next, there is a description of the available Picks on the **Scripts** tab.

1.3.1 Open Screen

Opens a specific Screen or Frame.



Settings for the Open Screen Pick

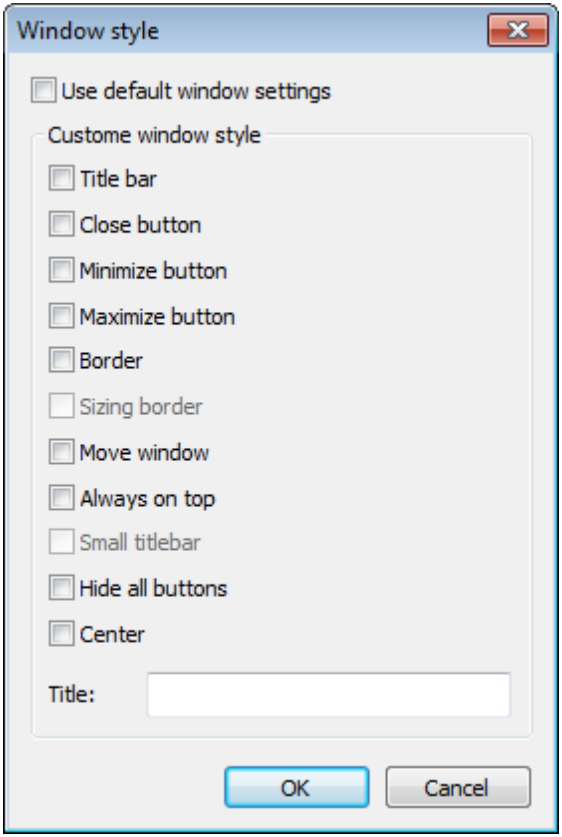
Available options for Open Screen Pick

OPTION	DESCRIPTION
Open Screen	Selects the Screen to open.
In Frame	Selects the Frame where the Screen will be viewed. If it is blank, it will be the main Frame (top).
Initial zoom	Defines the Screen's zoom, when displayed.

OPTION	DESCRIPTION
Parameter	Indicates a parameter to pass when calling the Screen.
Enable scrollbar	Allows using scroll bars on the Screen.
Specify screen position	Indicates the Screen position, in pixels.
Specify screen size	Indicates the Screen size, in pixels or Himetric.
Window style	Opens the **Window Style** dialog box.

1.3.1.1 Window Style Dialog Box

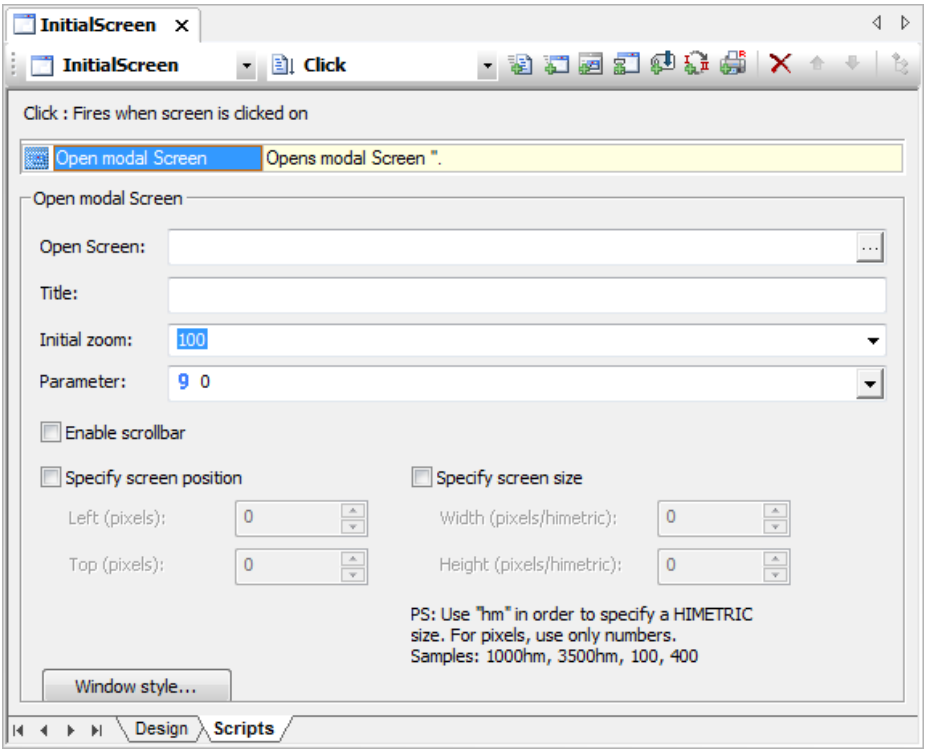
Allows users to set the style of the window to display, by defining title and availability of borders, and the close, maximize and minimize buttons, among other settings. If the **Use default window settings** option is selected, the system disables these window options and assumes the Viewer's default style, as seen on **Viewer** tab, under Viewer properties.



Window Style dialog box

1.3.2 Open Modal Screen

Opens a Modal Screen, that is, a Screen that does not allow user interaction with other Screens while it is active.



Settings for the Open Modal Screen Pick

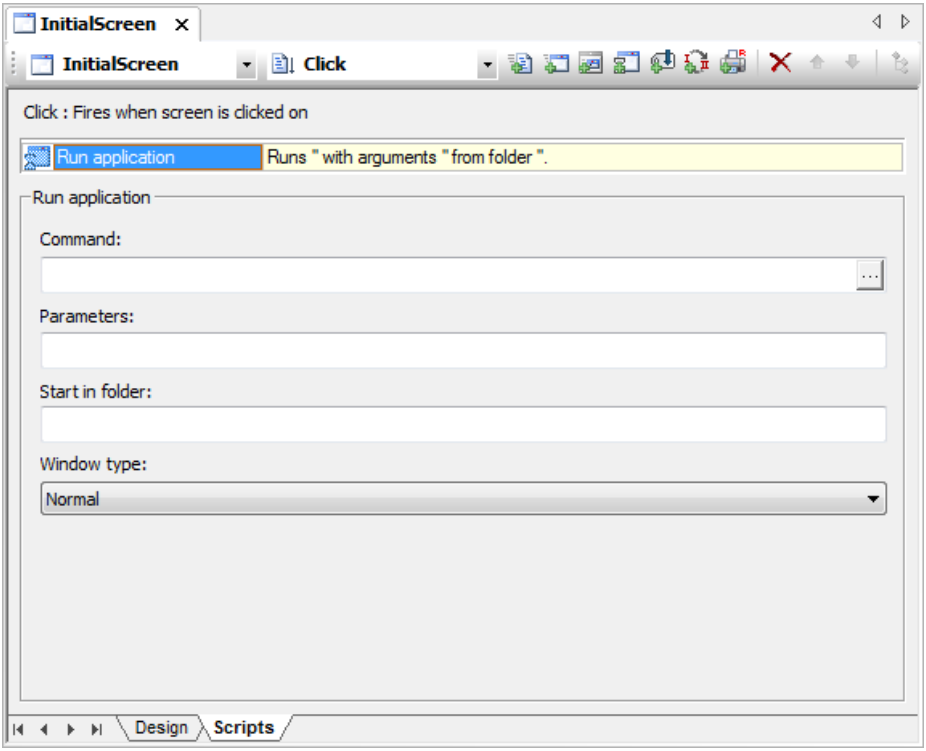
Available options for Open Modal Screen Pick

OPTION	DESCRIPTION
Open Screen	Selects the Screen to open.
Title	Defines the window title. This text is concatenated to the Screen's name.
Initial zoom	Defines Screen's zoom, when displayed.
Parameter	Indicates a parameter to pass when calling the Screen.
Enable scrollbar	Allows using scroll bars on the Screen.
Specify screen position	Determines the frame's position, in pixels, on the Screen, starting at Screen's upper left side.
Specify screen size	Determines Screen's width and height, in pixels or Himetric.

OPTION	DESCRIPTION
Window Style	Allows configuring the style of the window to display, by defining title and the availability of borders and the close, maximize and minimize buttons, among other settings (check the **Window Style Dialog Box** topic).

1.3.3 Run Application

Executes a specific application.



Settings for the Run Application Pick

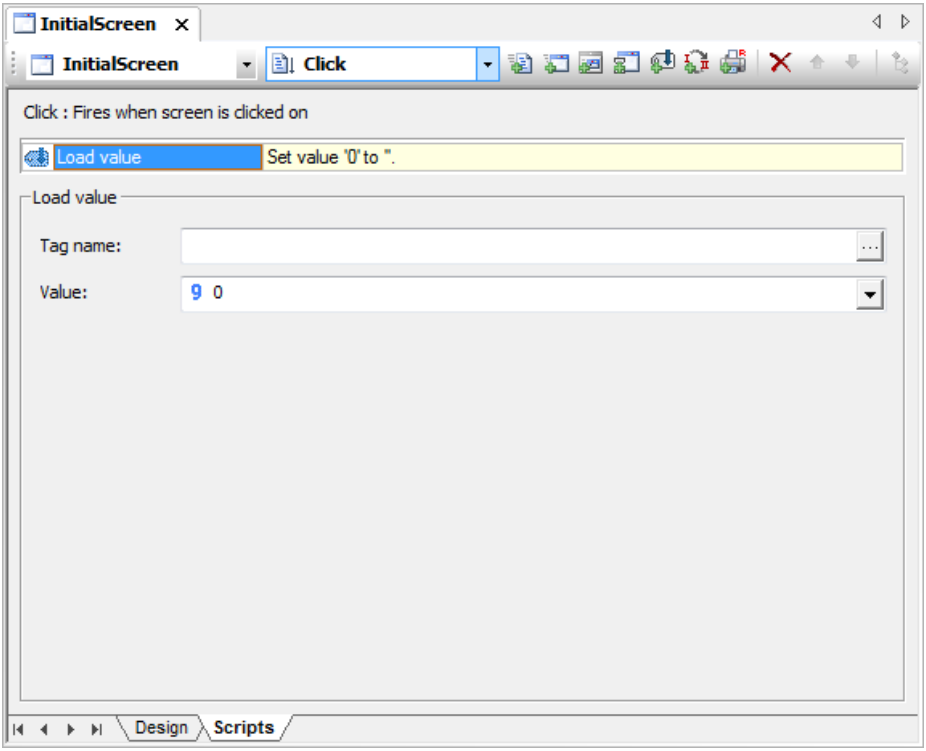
Available options for Run Application Pick

OPTION	DESCRIPTION
Command	By clicking ..., it is possible to browse the disk to select an application file to run.
Parameters	Allows specifying arguments for application call.
Start in folder	Determines the working directory of the application to run.

OPTION	DESCRIPTION
Window type	Determines the type of the execution window for the application: **Normal**, **Minimized**, or **Maximized**.

1.3.4 Load Value

Loads a value to a Tag.



Settings for the Load Value Pick

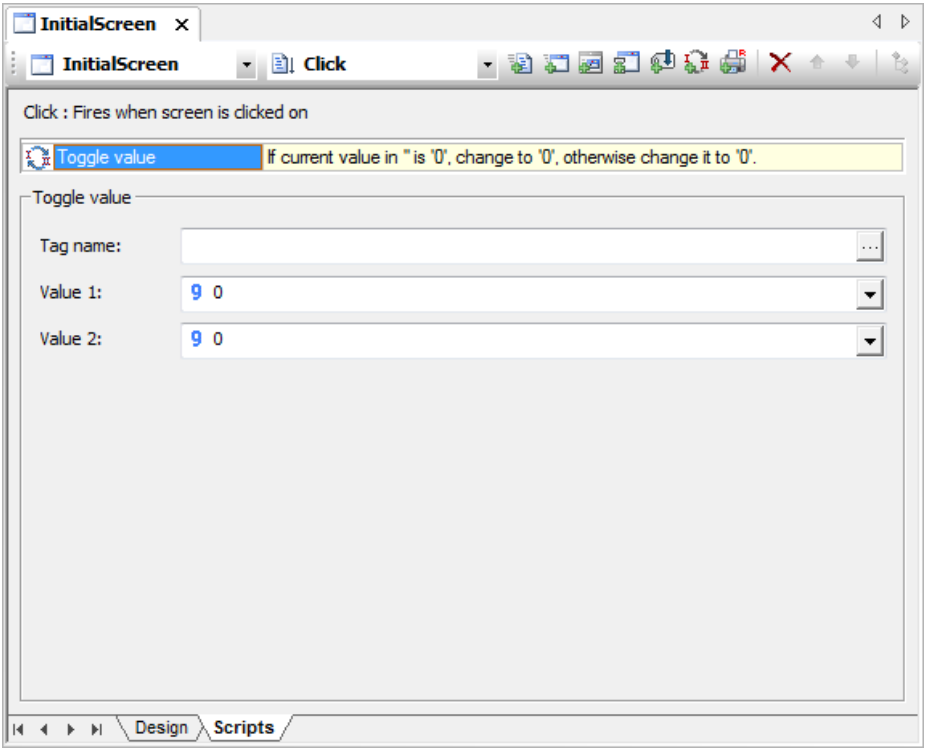
Available options for Load Value Pick

OPTION	DESCRIPTION
Tag name	Specifies the Tag name to which the value will be loaded. It is possible to select the Tag in the AppBrowser, by clicking ...
Value	Determines the value to load to the Tag. It is possible to select a data type, by clicking the arrow key.

1.3.5 Toggle Value

Allows toggling a Tag value. If the Tag value is equal to *Value1*, then the Tag receives *Value2*. If the Tag value is equal to *Value2*, then the Tag receives *Value1*. If the Tag value is different from either *Value1* or *Value2*, then the Tag receives *Value1*.

It is possible to add as many **Toggle Value** Picks as needed. This allows checking multiple values for the same Tag, or even for several Tags in the same event.



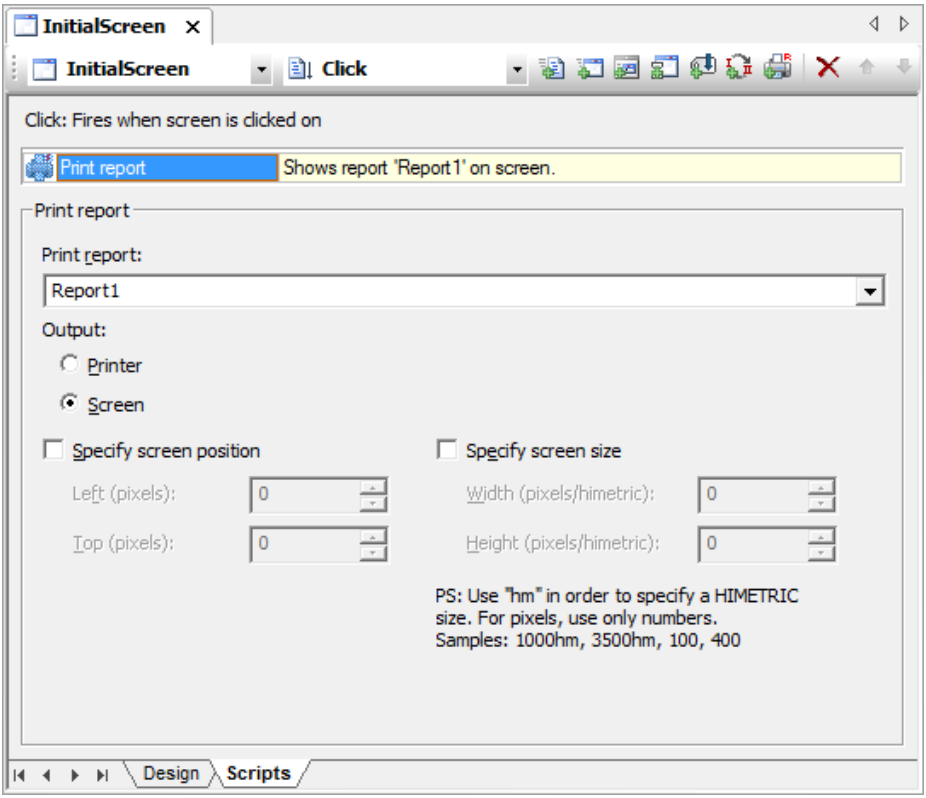
Settings for the Toggle Value Pick

Available options for Toggle Value Pick

OPTION	DESCRIPTION
Tag name	By clicking ..., the AppBrowser is opened to select the preferred Tag.
Value 1	Determines the first value to compare. If the Tag value is equal to *Value1*, then the Tag receives *Value2*.
Value 2	Determines the second value to compare. If the Tag value is equal to *Value2*, then the Tag receives *Value1*.

1.3.6 Print Report

Allows printing a Report to a printer or on screen.



Settings for the Print Report Pick

Available options for Print Report Pick

OPTION	DESCRIPTION
Print report	Allows selecting a Report to print.
Output	Determines a Report's output type: • **Printer**: Sends this Report to a printer. Corresponds to the **Print** method • **Screen**: Performs a printing preview of this Report on screen. Corresponds to the **PrintPreview** method
Specify screen position	Determines the position of the printing preview, in pixels, starting from the upper left corner of the screen. If these values are not defined, the printing preview is displayed at position (0, 0).

OPTION	DESCRIPTION
Specify screen size	Specifies the size of the printing preview on screen, in pixels or Himetric. If these values are not defined, the printing preview is created with 500 x 500 pixels and the window is opened maximized. If only one of these dimensions is defined, width or height, the other dimension is configured with 500 pixels and the window is not opened maximized.

1.4 User-Defined Events

Although E3 comes with a wide range of events, there are times when users may want to create a specific event for their applications.

An example of using user-customized events would be a calculation or a more complex task in an object, when the generated event comes from another Tag or property.

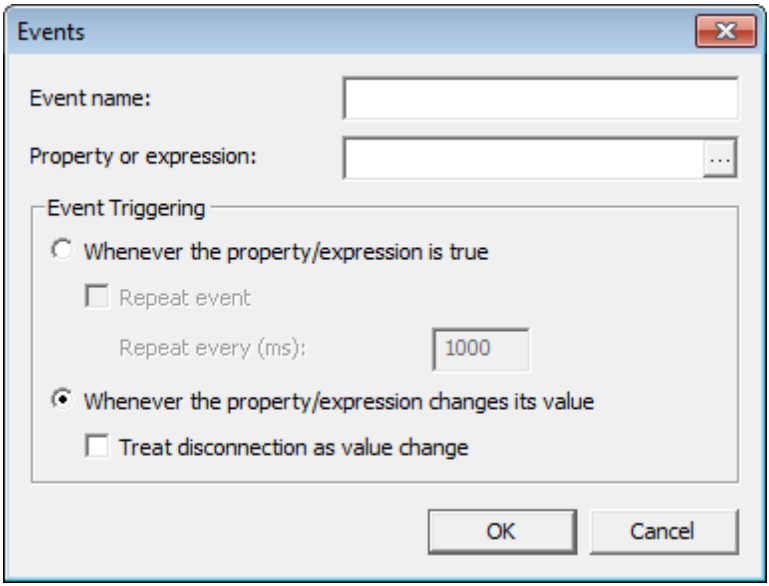
Although users can create and execute this type of work from the Tag or property generating that event, there are several advantages in keeping the script next to the object that suffers this script's action. Among these, there is the additional work needed to point an object to another one, not to mention the small amount of work needed in maintenance, because if by any reason it is necessary to modify or delete a Tag or property generating this event, it is not necessary to modify a second object.

Another advantage comes from the fact that if users erase the Tag generating this event, the object does not lose its script, as long as users indicate another generating source for this event.

The generation of internal events also makes the creation of libraries easier, because each time a library component is inserted in an application, it also brings scripts and calculations that can be necessary, reducing configuration work.

To generate a new internal event in an object, follow these procedures:

1. Click **Create user event** or click **Create new event** on the Events List. E3 then opens a window to define event properties.



Window for adding user-customized events

Available options for Events window

OPTION	DESCRIPTION
Event name	Name that identifies this event.
Property or expression	Expression generating this event. It can be copied using AppBrowser, by clicking ...
Whenever the property/expression is true	Indicates whether this event is an **etOnEvent**-type (this event occurs whenever the expression is true) or an **etWhileEvent**-type (this event occurs cyclically, at pre-defined intervals).
Repeat event	When filled in, indicates that it is an **etWhileEvent**-type event. Indicates this event's repetition cycle, that is, its periodicity while the generating expression is true, in milliseconds.
Whenever the property/expression changes its value	Indicates that it is an **etOnValueChanged**-type event, that is, this event occurs whenever the generating expression changes its value.
Treat disconnection as value change	Indicates whether a connection or disconnection of the expression generating this event must be considered as a change.

2. Click **OK** to finish the process and insert this event. It now appears on the Events

list.

3. To change this event, select it and click **Edit user event** . The previous window is then opened again to edit event's data.

4. To delete this event, select it and click **Remove user event** .

IMPORTANT: When clicking **Remove user event**, scripts for this event are lost.

CHAPTER 2

Programming in E3

Although the majority of VBScript aspects apply to scripts programming with E3, some particularities must be emphasized regarding the implementation of the concept of object orientation on the system.

VBScript is a language based on Visual Basic that brings the capacity of scripting to applications that run on Windows.

VBScript exchanges information with applications using the *ActiveX Scripting* technology. With ActiveX Scripting, browsers and other client applications, such as Viewers, can compile scripts and call functions, among other procedures. This enables scripts developed for an application or library, which can be executed in a graphical interface, to run either in Viewer or in an Internet browser, with no need to adapt it on the application.

Further information about VBScript can be obtained on the **VBScript Reference Guide**, in the Program Group **Elipse E3**.

2.1 Programming Environment

The programming environment for scripts in Studio can be accessed by right-clicking any object, and then selecting the **Properties** option. On **Scripts** tab of the object view, users can see a combo box where they can define which event will be the script's generator. As seen in the previous chapter, there are two types of events in an E3 object: pre-defined events, and user-customized events.

Pre-defined events vary from object to object, depending on its usage and functionality. A Screen object, for example, has graphical-interface-related events, such as **Click** (called when clicking the object) or **DbClick** (called when double clicking it); an object such as an I/O Driver, on the other hand, has communication-related events, such as **OnCommError** (called when there is a communication error). Users can also define other events for the object, as seen before.

When users link a script to an event in an object, the text field presents a procedure declaration whose definition is automatic, and formed by the following text:

```
Sub ObjectName_EventName()
End Sub
```

Where `ObjectName` is the name of the linked object, and `EventName` is the name of the aforementioned event. Script commands should be placed between these two lines.

To help typing the script, users can use the AppBrowser. After choosing the preferred method or property, they can click **Copy**. The chosen Tag, property, or method will be

inserted at the cursor position on script's edit view. The cursor position is shown by a blinking insertion point.

2.2 Declaring Variables

Users can declare variables in two different ways: **implicitly** or **explicitly**.

To declare a variable implicitly, simply use its name on the script. The variable will automatically be created and initialized with the attribution value, or it will remain with the EMPTY value, in case it does not receive any value before using it.

This is a quick practice, but if the script is very large, it may cause confusion and creation of more than one variable with the same name, generating bugs in the script.

To declare variables explicitly, use the **Dim** command, as in the next example:

```
Dim Temperature
```

Users can declare multiple variables by separating each variable's name by a comma. For example:

```
Dim Left, Right, Top, Bottom
```

Because all scripts in E3 are linked to a specific object, variables are always local, valid only on script's scope. For public or global variables, users must create an Internal Tag and then use it to store the preferred value.

2.3 Getting References to Objects

One of the most important features to consider when working with scripts inside E3 is the separation between processes executed in the server and those executed in the client's interface (Viewer). To work with scripts, users can manipulate:

- Server objects via server
- Server objects via Viewer(s)
- Viewer objects via the same Viewer

However, users cannot directly manipulate:

- Viewer objects via server (this is only possible by creating events in Viewer, which are connected to variables in the server)
- Viewer objects via another Viewer (this is only possible by creating events connected to variables in the server)

Such limitations come from the fact that, by definition, there is an independence between what the Viewer's station is performing or viewing and the server, and vice

versa. So, all activities, both in the server and in Viewer, need to be coordinated either in an asynchronous way or through events, to operate harmoniously.

Thus, due to this independence, when creating a script, first users should obtain a correct reference to the objects they want to manipulate, that is, it is necessary that the object be found on several E3 modules first.

Remember that when users edit a script, they can use the AppBrowser, which allows the path of a method or property to be completely copied to the script, thus helping to create scripts.

Therefore, to access external objects being manipulated in a script, some basic guidelines are used. For example, if users want to manipulate the value of an I/O Tag, the path is **Server - Driver - Folder (if available) - Tag**. But if their goal is to manipulate a button on a Screen, the path is **Viewer - Frame (if available) - Screen - Button**.

There are basically three source locations for scripts, according to the methodology for accessing objects:

- Server
- Screens and Frames (Viewer)
- ElipseX (libraries): they may be XObjects (running on the server) or XControls (running on a Viewer)

2.3.1 Accessing Server Properties

To access an object running on the server from a Screen object or an ElipseX, use the **Application.GetObject** method.

The word **Application** returns the application object related to the current context of the object, and the **GetObject** method searches for an object with the given name within **Application**, on the server. Example:

```
Sub Button1_Click()
  Application.GetObject("Driver1")._
    Item("tag001").AllowRead = False
End Sub
```

Or else:

```
Sub Button1_Click()
  Application.GetObject("Driver1.tag001").AllowRead = False
End Sub
```

The **Item** method was used to locate `tag001` from a reference to `Driver1`, because the Driver is a collection of Tags. After locating the object, its properties and functions can be freely accessed.

In case users need to accomplish another operation with `Driver1` or `tag001`,

another alternative for the previous script is:

```
Sub Rectangle1_Click()
  Set obj = Application.GetObject("Driver1")
  obj.Item("tag001").AllowRead = False
  obj.Item("tag002").AllowRead = False
End Sub
```

In this case, the variable `obj` is pointing to the object `Driver1`, and the next time users want to access some object descending from `Driver1` inside the script, they can use the variable `obj` directly. This brings a performance enhancement, since every call to the **GetObject** method accesses the server only once. With this technique, users avoid unnecessary calls to a server. This example uses the **Set** command, explained later. Notice that the usage of variables also makes code clearer and more easily modifiable. In case it is necessary to change the object where users want to execute commands, just change the attribution line of this variable.

The word **Application** in scripts can indicate either functions executed on a Viewer or on the Server. In this case, the Application object knows beforehand which functions should be executed in each case. However, users cannot execute Viewer functions inside the server, and it is not possible to execute server functions inside a Viewer.

2.3.2 Accessing Studio Properties

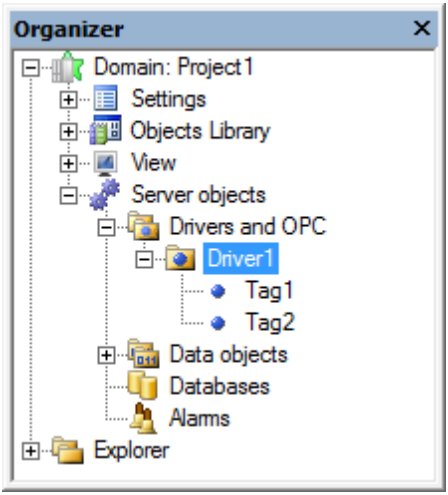
Accessing any server object in a script that runs in Studio can be performed by using the **Application.GetObject** directive. The word **Application** returns the application object related to the current object's context, and the **GetObject** method searches on the load Domain in Studio for a server object with the provided path. Example (the **CustomConfig** event is triggered in Studio):

```
Sub XControl1_CustomConfig
  Application.GetObject("Data.DemoTag1").DocString =
"Documentation"
End Sub
```

2.3.3 Accessing Server Properties from the Server

Take the example of accessing Tag properties from another Tag, origin and destination are on the server, even though connected via a `Driver1`'s parent module.

In this case, the **Parent** declaration must be used. This allows first accessing the parent object where the script is, and then users traverse the hierarchy to look for another element.

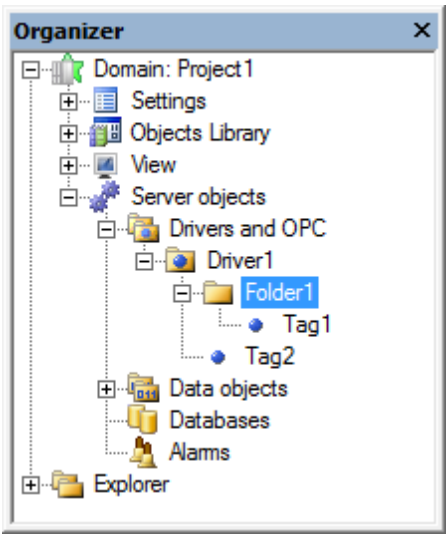


Driver1 is the parent object of Tag1 and Tag2

Example:

```
Sub Tag1_OnRead()
  Parent.Item("Tag2").AllowRead = False
End Sub
```

If users are inside a group and they want to access the same Tag2, they can nest various **Parent** commands.



Folder1 is the parent object of Tag1

Example:

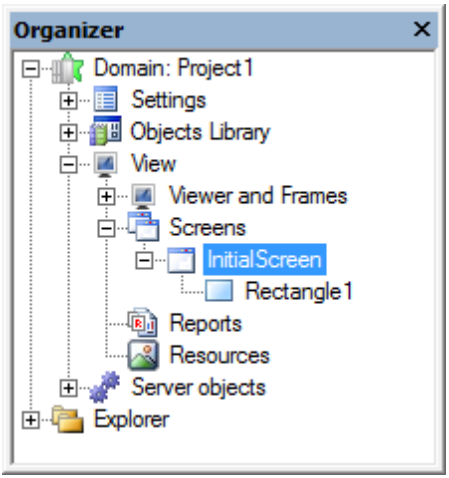
```
Sub Tag1_OnRead()
```

```
  Parent.Parent.Item("Tag2").AllowRead = False
End Sub
```

2.3.4 Accessing Screen Objects from a Screen Script

In this case, users must only use the **Item** method, as the objects are children of the Screen. For example:

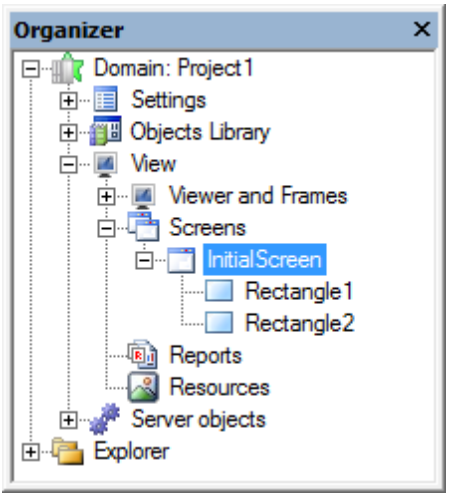
```
Sub Screen1_OnPreShow(vArg)
  Item("Rectangle1").Visible = True
End Sub
```



Rectangle1 is an item of InitialScreen

2.3.5 Accessing Screen Objects from a Script on Another Object Screen

In this case, users can use the **Parent** or the **Screen** method.



InitialScreen is the parent object of Rectangle1 and Rectangle2

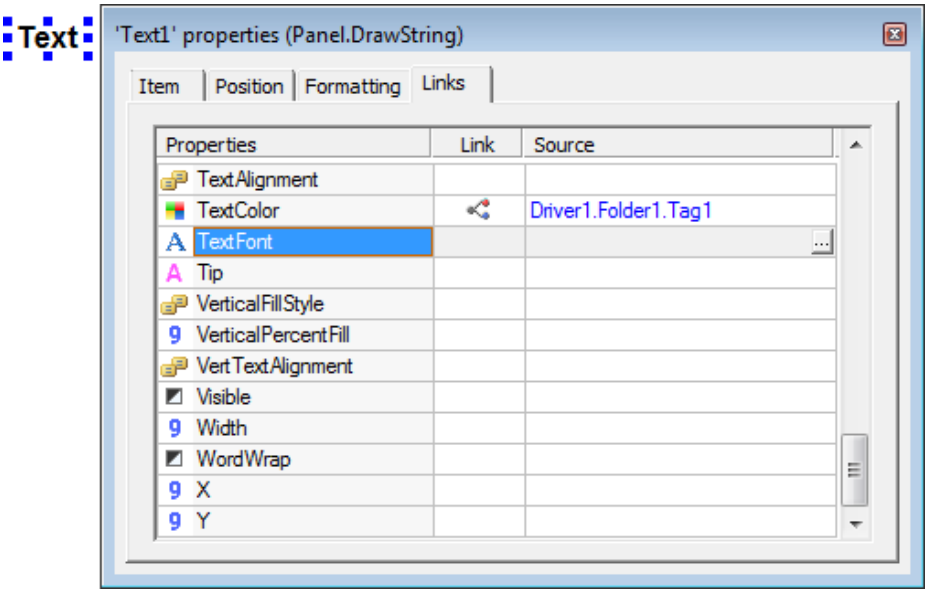
Example:

```
Sub Rectangle1_Click()
  Parent.Item("Rectangle2").Visible = True
End Sub
```

2.3.6 Changing a Screen or Screen Objects from the Server

The modification of any behavior on a Screen can be performed using Links (the server automatically reports all changes in the chosen variables to Viewers), or by an explicit Viewer's search for information in the server. The whole graphical interface linking operation is accomplished from client to server and not from server to client. Thus, it is not possible to change Screens or objects from the server by using scripts, because each data client is a different copy of the Screens.

A practical example is either to change the color of a text in the Screen to green when a Tag is turned on (value one) or to red when it is turned off (value zero). In this case, simply create a Digital Link between `Text1`'s **TextColor** property and `Tag1`. Links are preferable due to execution speed and also simplicity in maintaining and building applications.

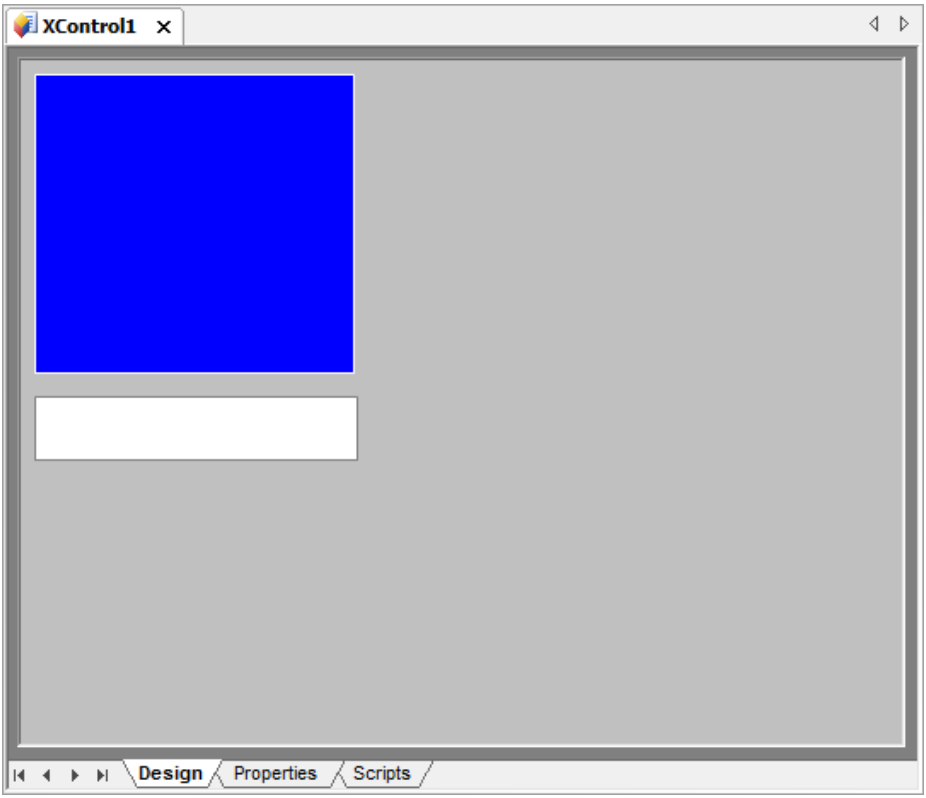


Linking text color to Tag1's value

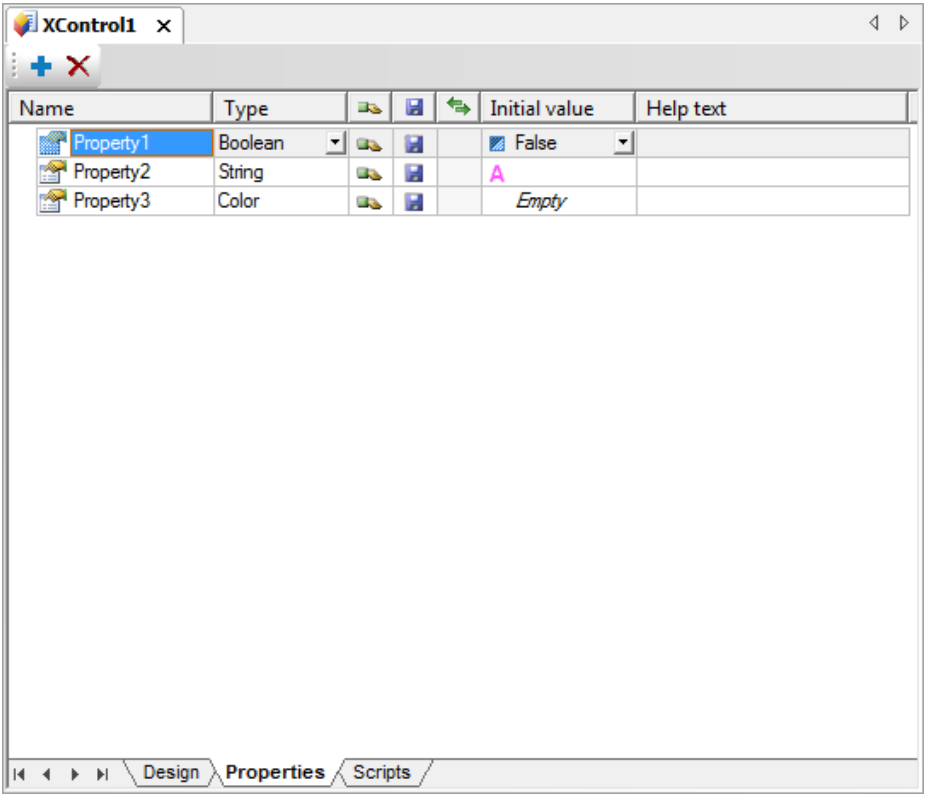
Another way to execute the previous procedure is by creating a script in the Viewer that constantly checks whether Tag1 changed its value, and then changes the text color. This type of script can be accomplished, but it strongly degrades application performance. Thus, this practice is **not** recommended.

2.3.7 Accessing ElipseX Objects from the ElipseX Itself

When creating an ElipseX, users can declare properties (XProperties) and insert objects, which can be Screen objects (XControls) or server objects (XObjects). To access XProperties via scripts, access the property's name directly.



Design tab



Properties tab

For example, in the previous figure there is an object XControl1 with the **Property1** property, and objects Text1 and Rectangle1.

The **Property1** property, which is a **Boolean**-type, can be accessed with the following script:

```
Sub XControl1_OnStartRunning()
  XControl1.Property1 = True
End Sub
```

Or else with the following script:

```
Sub XControl1_OnStartRunning()
  Property1 = True
End Sub
```

If the ElipseX has internal objects, then it is possible to use the **Item** method to get a reference for these objects, as in the following script:

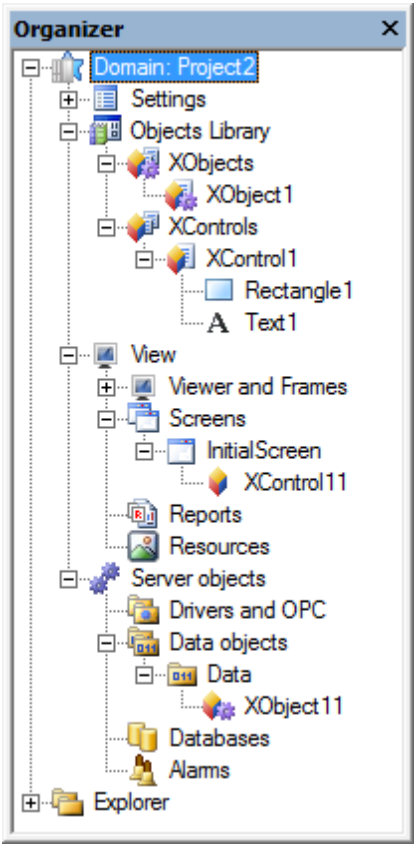
```
Sub XControl1_OnStartRunning()
  Item("Text1").Value = "engine"
  Item("Rectangle1").ForegroundColor = RGB(212, 208, 20)
```

```
End Sub
```

2.3.8 Externally Accessing Objects of an ElipseX

An ElipseX object can only be accessed externally via its properties, by using its created instances. It is not possible to access internal objects directly.

If an ElipseX is an XControl, it behaves as a Screen object. For example, consider the application on the next figure.



XControl (example)

To change XControl's **Property1** property, users can write the following script on a Button's **Click** event:

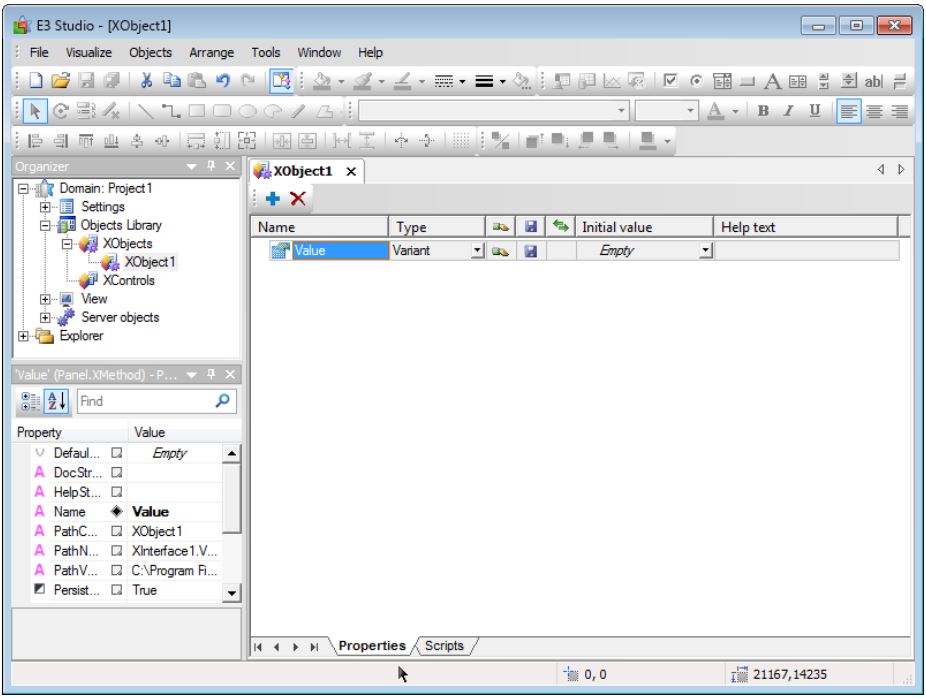
```
Sub CommandButton1_Click()
  Screen.Item("XControl11").Property1 = True
End Sub
```

Or else:

```
Sub CommandButton1_Click()
  Parent.Item("XControl11").Property1 = True
```

```
End Sub
```

In case of an XObject, insert it in a Data Server.



XObject (example)

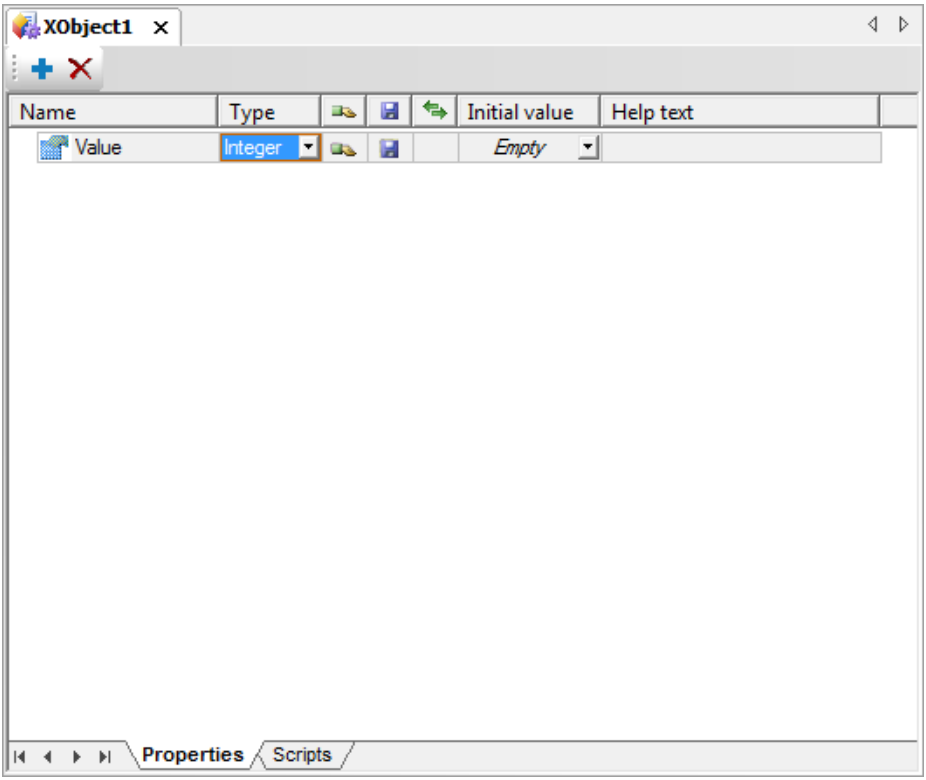
A possible script to alter XObject's **Value** property would be:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.XObject11").Value = 123
End Sub
```

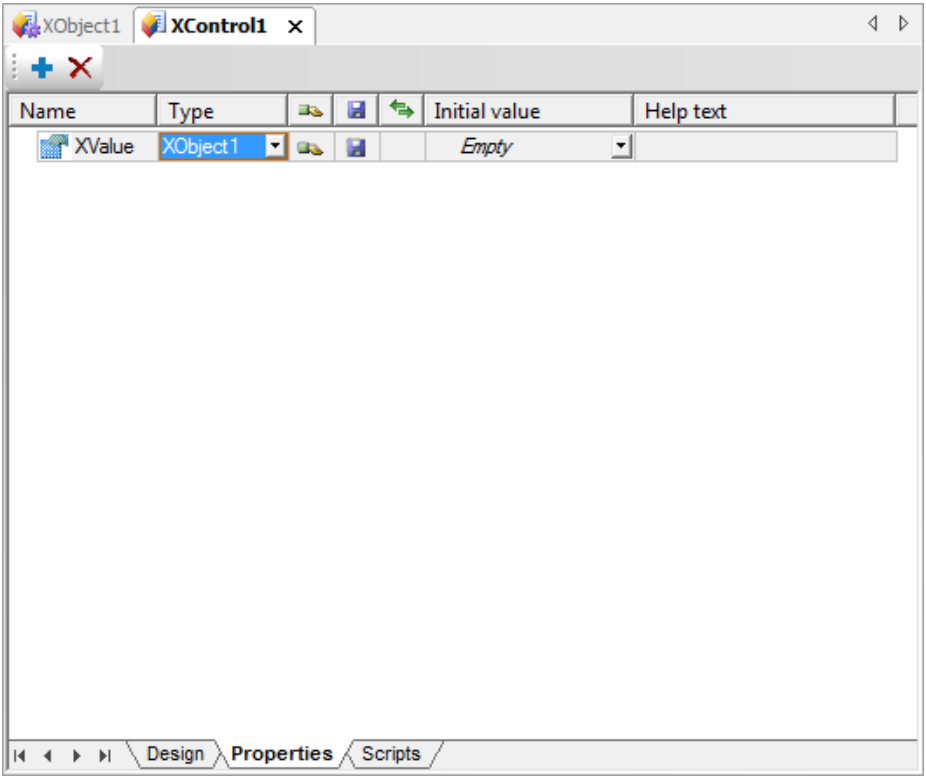
Or else:

```
Sub CommandButton1_Click()
  Application.GetObject("Data").Item("XObject11").Value = 123
End Sub
```

Users can also have an XControl accessing an XObject through an XProperty. For example, the next figure shows an XControl called "XControl1" with an **XValue** property. This property's type is **XObject1**, which is also the name of the XObject created.

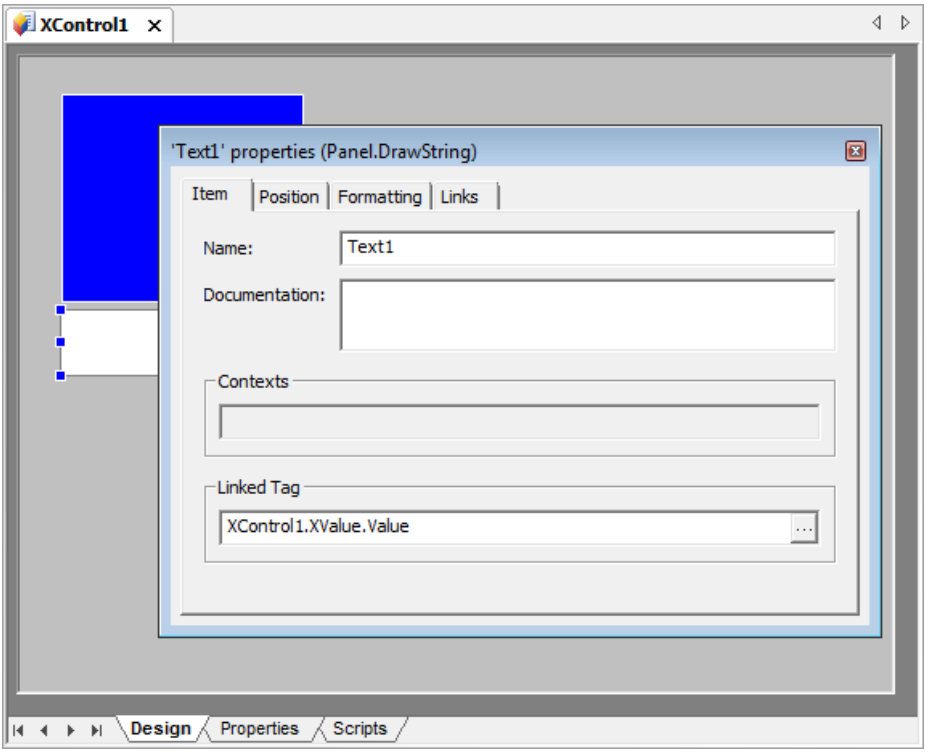

XObject1
Name
Type
Initial value
Help text
Value
Integer
Empty
Properties
Scripts

XObject



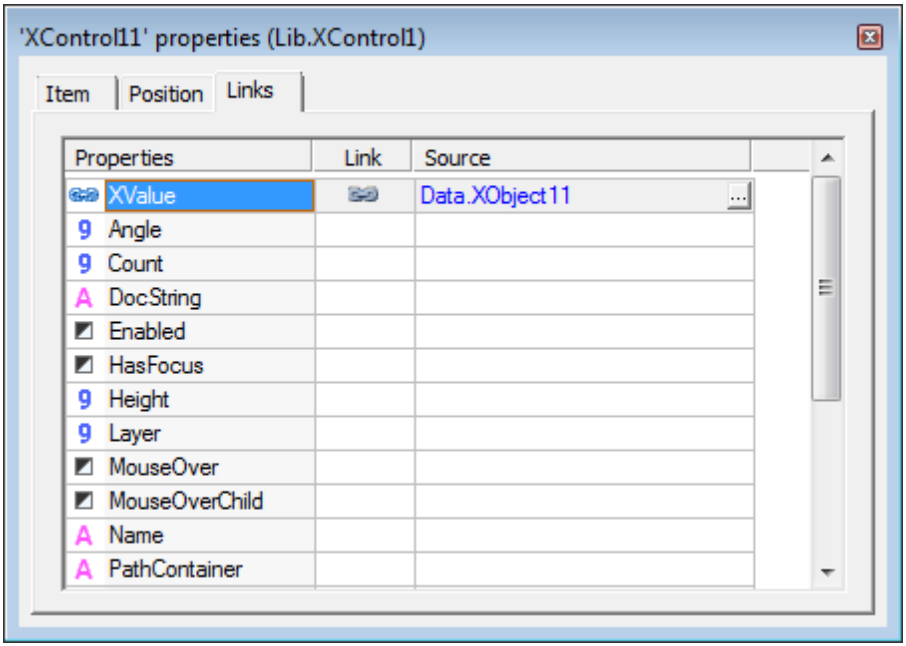
XControl

For example, users can link the value of object "Text1" with XObject1's **Value** property. This is performed via **XValue** property, created in "XControl1". Thus, the value of XObject1's **Value** property is displayed in object "Text1" of "XControl1".



Value property

In this project, the link of an "XObject11" instance can be linked to an "XControl11" instance using a Link in the **XValue** property.

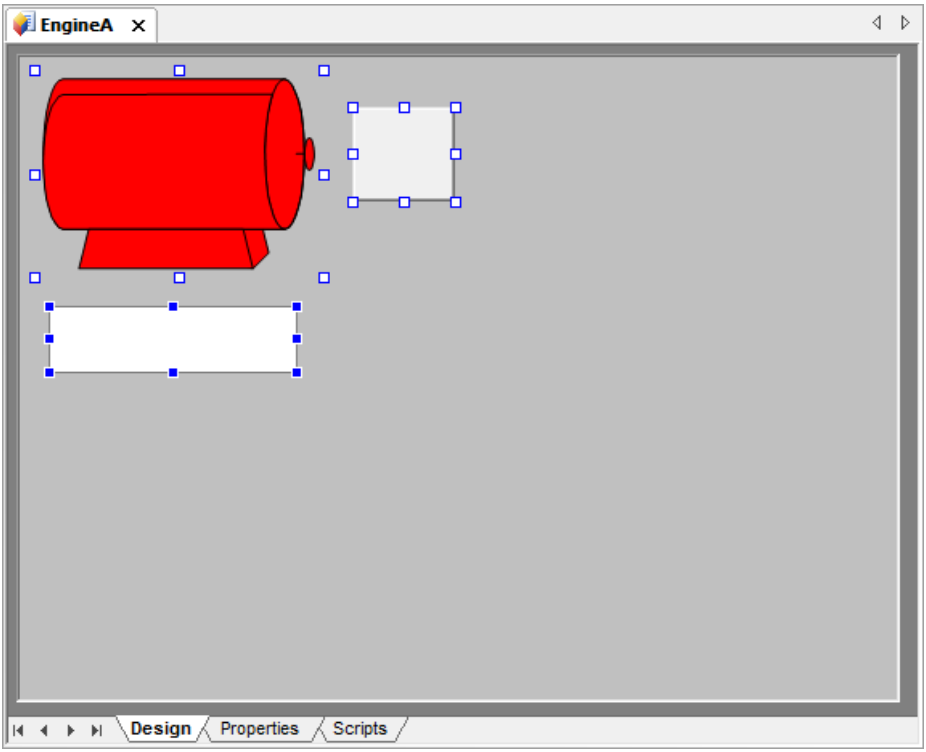


XValue (Link)

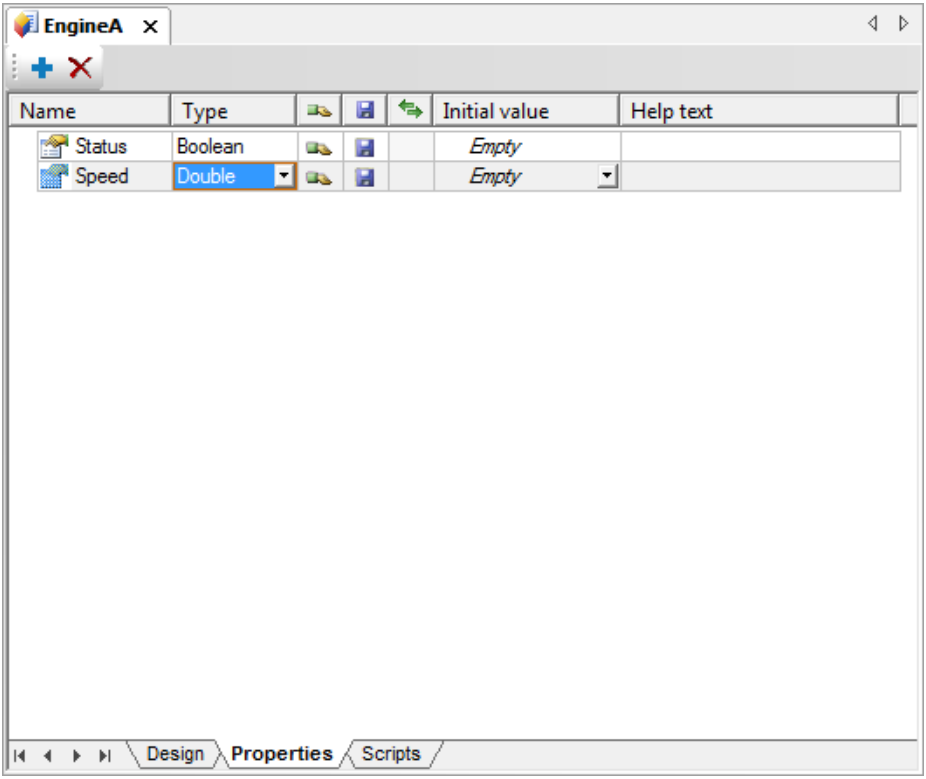
2.3.8.1 Example of Creating an ElipseX

Suppose a given application needs to supervise and command 10 engines. Each engine must be represented by a drawing on Screen, which displays a green color when operating, and a red color when turned off. It must also display the engine command on Screen, sending instructions for turning it on and off, and its speed must also be displayed.

One possibility is creating an XControl called EngineA, with properties **Status** set to **Boolean** and **Speed** set to **Double**, as seen on the next figures:



Design tab

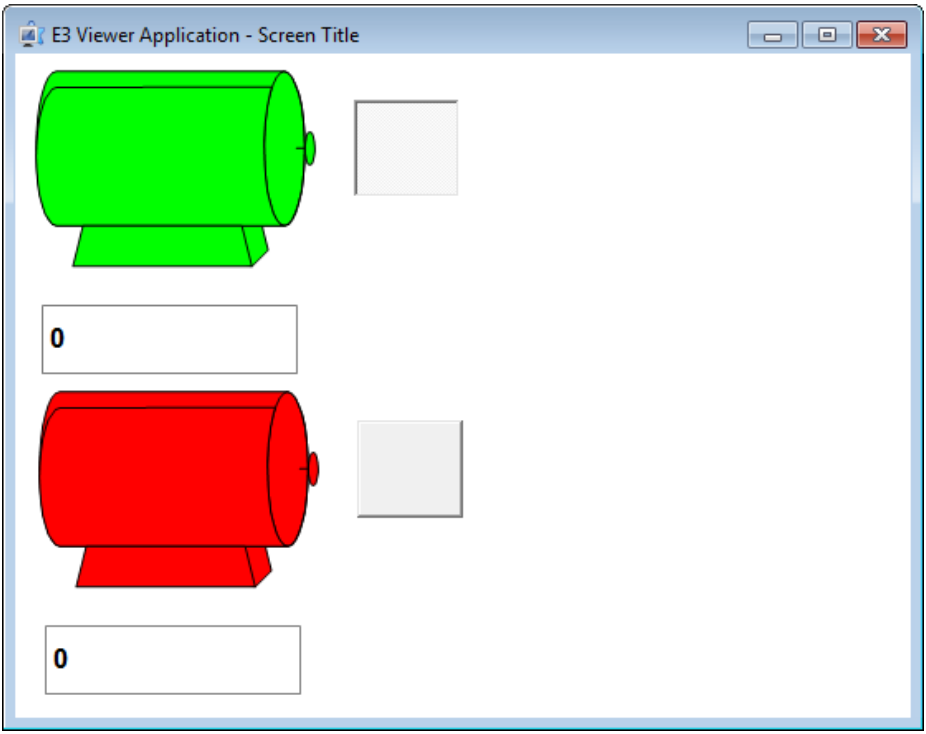


Properties tab

1. To indicate color, engine's **OverrideFillColor** property must be linked to XControl's **Status** property, by using a Digital Link. Set the **OverrideFillMode** property to **2 - SolidFill**.
2. To display speed, Display's **Value** property must be linked to XControl's **Speed** property.
3. The Toggle Button switches the **Status** property value via a Simple Link.

Notice that:

- Links within the library are internal, and their format is **Control_Name.Property_Name**
- The object, after being inserted on a Screen, must have these properties linked to real Tags, for each engine
- A Tag Link to the **Status** property must be performed for each EngineA

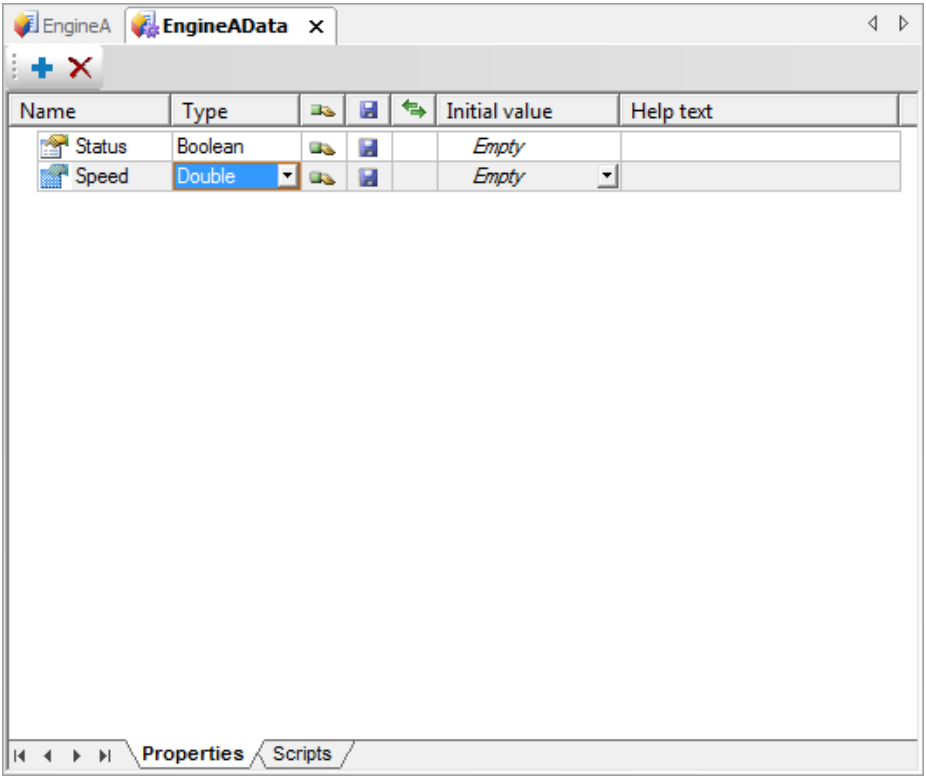


Viewer

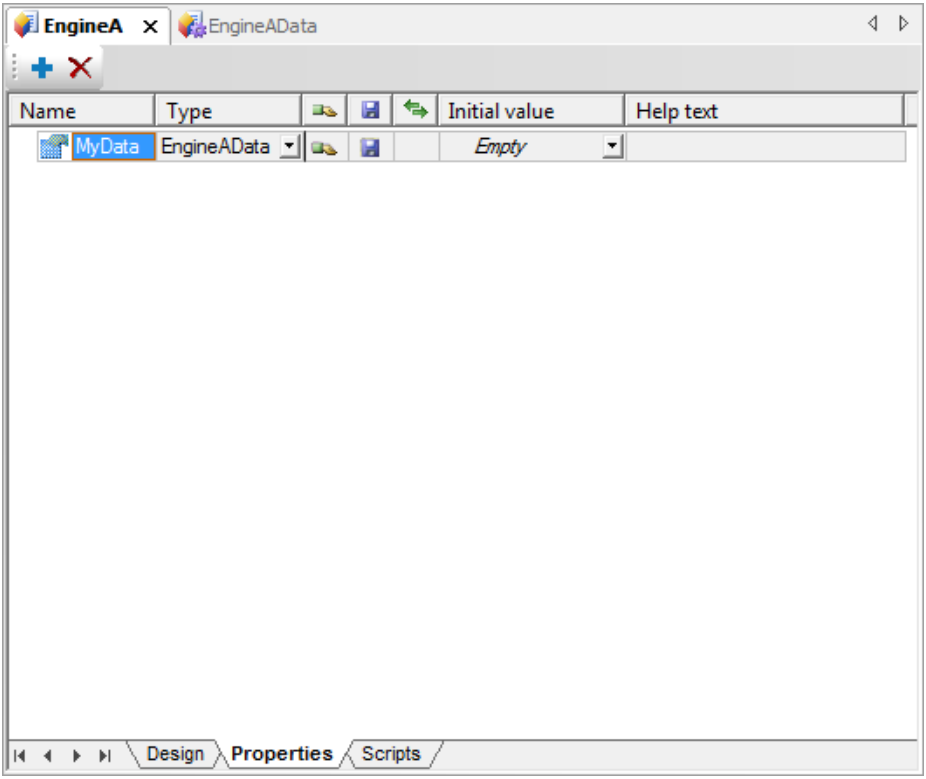
Another broader possibility is to use an XObject for the engine. Thus, all information regarding the engines remains in objects in the server. Hence, it is possible to create several types of interface for the engine (XControls), which brings only relevant information from the server, via XObject.

Then, the EngineA object would have to be modified to point to an XObject, instead of declaring all properties on itself.

1. Create an XObject called EngineAData, and declare **Status** and **Speed** properties in it.
2. Create an XControl EngineA with a single property, called **MyData**, of type **EngineAData**.
3. EngineAData must be inserted in a Data Folder in the server, corresponding to each engine. EngineA, by its turn, will point to the EngineAData needed, with no need to create new Tags.



Configuration of XObject's view



Configuration of XControl's view

4. The **Status** property, linked to engine's **OverrideFillCollor** property, is then **EngineA.MyData.Status**.
5. The **Speed** property, linked to the Display, is then **EngineA.MyData.Speed**.

2.4 Accessing Objects

Following the object oriented programming encapsulation concept, methods and properties are linked to their original objects. This means that users always have to indicate the object from which they are accessing a method or property.

2.4.1 Properties

For references to object properties, users must use the **GetObject** method. Its syntax is the following:

```
Application.GetObject("<object>").<property>
```

Where *<object>* is the name of the object, and *<property>* is the desired property.

Example:

```
Application.GetObject("Data.TempTank2").Type
```

For an easy typing, it is always advisable to use the AppBrowser, which already retrieves the correct syntax.

2.4.1.1 Value Property

In E3, many objects have a common property, called **Value**. In this specific case, users can access this property by using the name of the object itself:

```
Button1 = False
```

Which is equivalent to:

```
Button1.Value = False
```

2.4.2 Methods

The following syntax exemplifies a method call that does not need parameters:

```
Application.GetObject("<object>").<method>
```

Where *<object>* is the specific object, and *<method>* is the desired method.

If the method accepts parameters, use the following syntax:

```
Application.GetObject("<object>").<method>(<parameter>)
```

Where *<parameter>* is the parameter to pass to the method. When there is more than one parameter, use commas to separate them.

If the method returns a result that users want to keep, then the parameters must be placed between parentheses:

```
<V> = Application.GetObject("<object>").<method>(<parameter>)
```

Where *<V>* is the variable that receives the method's result.

2.5 Working with Collections

A collection is an object that manages a set of similar objects. The objects contained in a collection are referred by indexes, similarly to array references. Users can add or remove individual objects from a collection, as seen in the following example:

```
Sub CommandButton1_Click()
  ' Add a pen to the E3Chart1 object
  Screen.Item("E3Chart1").Pens.AddPen "Pen"
End Sub

Sub CommandButton2_Click()
  ' Removes the first pen
```

```
  Screen.Item("E3Chart1").Pens.Remove 0
End Sub
```

NOTE: The first object in a collection has index 1 (one).

All collections have a common property named **Count**, which is the number of existing objects (or children). Example:

```
Sub CommandButton1_Click()
  ' Shows a dialog box with the number of pens
  MsgBox Screen.Item("E3Chart1").Pens.Count
End Sub
```

2.5.1 Accessing Objects using the Item Method

Every collection has an **Item** method, which can be used to access any object within the collection. The **Item** method accepts an *Item* parameter, which can be a number (a positive integer) or the name of the object to access within the collection. The following examples adjust the E3Chart object's second Pen color:

```
Sub CommandButton1_Click()
  ' Changes the color of the third pen
  Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(212, 208, 20)
End Sub
```

Or else:

```
Sub CommandButton1_Click()
  ' Changes the color of a pen named "Pen2"
  Screen.Item("E3Chart1").Pens.Item("Pen2").Color = RGB(212,
208, 20)
End Sub
```

The previous commands are equivalent, the first one indicating the Pen's index within the collection, and the second one indicating its name.

2.6 Set Command

VBScript implements the concept of polymorphism from object-oriented languages, allowing a **Variant**-type variable to assume the form of any object via the **Set** command. This way, a variable works as a pointer to an object, allowing the access to its methods and properties. Example:

```
Sub CommandButton1_Click()
  Set E3Chart = Screen.Item("E3Chart1")
  E3Chart.Pens.Item(2).Color = RGB(212, 208, 20)
End Sub
```

In this example, the same task of the previous example has been accomplished, but the part related to getting the specific object was omitted. Without the **Set**

command, the same call must be written as:

```
Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(212, 208, 20)
```

Apparently, there is no advantage in this case, because users will be able to perform that with a single line of code. But, if they want to perform other operations in the same script right after, the process would be simpler and faster if the call to the **Item** method is not in all lines.

```
Sub CommandButton1_Click()
  ' A bad example:
  Screen.Item("E3Chart1").Pens.Item(0).Color = RGB(212, 208, 20)
  Screen.Item("E3Chart1").Pens.Item(1).Color = RGB(200, 208, 20)
  Screen.Item("E3Chart1").Pens.Item(2).Color = RGB(100, 208, 20)
End Sub

Sub CommandButton1_Click()
  ' A better example:
  Set Pens = Screen.Item("E3Chart1").Pens
  Pens.Item(0).Color = RGB(212, 208, 20)
  Pens.Item(1).Color = RGB(200, 208, 20)
  Pens.Item(2).Color = RGB(100, 208, 20)
End Sub
```

2.7 E3Globals

E3Globals is an E3 module that contains **methods** for global usage. The **GetCurrentWindowsUserName**, **GetLocalTime**, and **GetLocalTimeUTC** methods cannot be used in Links, only in scripts. The rest of the methods can be used in Links, as well as in scripts.

The way of using these methods admits two syntaxes, `E3Globals.<method>` or simply `<method>`, except for the Report object, where the `E3Globals.<method>` syntax is mandatory.

2.7.1 Methods

This section contains information about the methods of the E3Globals module.

2.7.1.1 BShl

BShl(*Value*, *Bits*)

Returns *Value* left-shifted the number of bits specified in *Bits*. This method returns an error in case the *Bits* parameter is outside the allowed range (from 0 to 31). This method is available in Links, as well as in scripts.

2.7.1.2 BShr

BShr(*Value*, *Bits*, *PreserveSign*)

Returns *Value* right-shifted the number of bits specified in *Bits*. This method returns an error in case the *Bits* parameter is outside the allowed range (from 0 to 31). The *PreserveSign* parameter is an optional **Boolean** which, if True, fills in the bits to the left with a copy of the signal bit. The default value of this parameter (False) fills in the bits to the left with zeroes. This method is available in Links, as well as in scripts.

2.7.1.3 Choose

Choose(*Index*, *Values*)

Returns one of the items specified in *Values*, based on *Index* (starts at zero). This method returns Null in case the value of *Index* is less than 0 or greater or equal to the number of values in *Values*. This method is available in Links, as well as in scripts.

NOTE: The **Choose** method does not propagate values' quality nor timestamp. If a Link contains the expression `Choose(TagIndex, Tag1.Value, Tag2.Value, Tag3.Value)`, the result is the chosen value, however with Good (192) quality and the current timestamp. In order to preserve this information, users must specify only the object, as for example `Choose(TagIndex, Tag1, Tag2, Tag3)`.

2.7.1.4 E3Format

E3Format(*Value*, *Format*)

Formats the expression in *Value* using the format specified in *Format*. This format uses the same definitions of the **Format** property of Text, Display, and SetPoint objects. This method is available in Links, as well as in scripts.

2.7.1.5 GetBit

GetBit(*Value*, *BitIndex*)

Returns the value (True or False) of the *Value* bit specified by *BitIndex*. This method returns an error in case the *BitIndex* parameter is outside the allowed range (from 0 to 31). This method is available in Links, as well as in scripts.

2.7.1.6 GetComputerName

GetComputerName()

Returns a **String** containing the name of the local computer. This method is available in Links, as well as in scripts.

2.7.1.7 GetCurrentWindowsUserName

GetCurrentWindowsUserName()

Returns a **String** containing the name of the user logged on the current process. This method is not available in Links, only in scripts.

2.7.1.8 GetLocalTime

GetLocalTime()

Returns the date and time of the local computer, with a precision of milliseconds and in the local time zone. This method is not available in Links, only in scripts.

2.7.1.9 GetLocalTimeUTC

GetLocalTimeUTC()

Returns the date and time of the local computer, with a precision of milliseconds and in the UTC (*Coordinated Universal Time*) time zone. This method is not available in Links, only in scripts.

2.7.1.10 IIf

IIf(*Condition*, *ExprTrue*, *ExprFalse*)

Returns the expression contained in *ExprTrue* if the condition evaluated in *Condition* is true, and the expression contained in *ExprFalse* if the condition evaluated is false. This method is available in Links, as well as in scripts.

NOTE: The **IIf** method does not propagate values' quality nor timestamp. If a link contains the expression `IIf(Tag1.Value == 0, Tag2.Value, Tag3.Value)`, the result is the value of Tag2 or Tag3, however with Good (192) quality and current timestamp. To preserve this information, users must specify only the object, as for example `IIf(Tag1.Value == 0, Tag2, Tag3)`.

2.7.1.11 OPCGetLimit

OPCGetLimit(*Quality*)

Returns the **Limit** information of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. Possible returned values for this method are:

- **0**: Free
- **1**: Low
- **2**: High

- **3**: Constant

2.7.1.12 OPCGetQuality

OPCGetQuality(*Quality*)

Returns the **Quality** information of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. Possible returned values for this method are:

- **0**: Bad
- **1**: Uncertain
- **2**: Not used
- **3**: Good

2.7.1.13 OPCGetSubStatus

OPCGetSubStatus(*Quality*)

Returns the **Substatus** information (from 0 to 15) of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts. The OPC Standard specifies the following values:

- **Good Quality**:
  - **0**: non-specific
  - **1**: local override
- **Bad Quality**:
  - **0**: non-specific
  - **1**: configuration error
  - **2**: not connected
  - **3**: device failure
  - **4**: last known value
  - **5**: communication failure
  - **6**: out of service
- **Uncertain Quality**:
  - **0**: non-specific
  - **1**: last usable value
  - **4**: sensor not accurate

- **5**: EU exceeded
- **6**: subnormal

2.7.1.14 OPCGetVendor

OPCGetVendor(*Quality*)

Returns the **Vendor Specific** information (from 0 to 255) of an OPC Quality specified by the *Quality* parameter. This method is available in Links, as well as in scripts.

2.7.1.15 OPCIsBad

OPCIsBad(*Quality*)

Returns True if the OPC Quality is Bad, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.16 OPCIsGood

OPCIsGood(*Quality*)

Returns True if the OPC Quality is Good, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.17 OPCIsUncertain

OPCIsUncertain(*Quality*)

Returns True if the OPC Quality is Uncertain, and False otherwise. This method is available in Links, as well as in scripts.

2.7.1.18 OPCMakeQuality

OPCMakeQuality(*QualityFlag*, *SubStatus*, *Limit*, *Vendor*)

Returns a new OPC Quality value, built using the values passed in the *QualityFlag*, *SubStatus*, *Limit*, and *Vendor* parameters. This method is available in Links, as well as in scripts. Possible values for each one of the parameters are:

- **QualityFlag**: Specifies the quality of the value
  - **0**: Bad
  - **1**: Uncertain
  - **3**: Good
- **SubStatus**: Specifies the substatus of the value (between 0 and 15, see the **OPCGetSubStatus** method for possible values). If this parameter is omitted, assumes 0 (zero)

- **Limit**: Specifies the limit of the value. If this parameter is omitted, assumes 0 (zero)
  - **0**: Free
  - **1**: Low
  - **2**: High
  - **3**: Constant
- **Vendor**: Specific value of the vendor (between 0 and 255). If this parameter is omitted, assumes 0 (zero)

2.7.1.19 OPCSetLimit

OPCSetLimit(*Quality*, *Limit*)

Changes the **Limit** information of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts. Possible values for the *Limit* parameter are:

- **0**: Free
- **1**: Low
- **2**: High
- **3**: Constant

2.7.1.20 OPCSetQuality

OPCSetQuality(*Quality*, *QualityFlag*)

Changes the **Quality** information of an OPC Quality and returns the modified value. This method is available in Links as well as in scripts. Possible values for the *QualityFlag* parameter are:

- **0**: Bad
- **1**: Uncertain
- **2**: Not used
- **3**: Good

2.7.1.21 OPCSetSubStatus

OPCSetSubStatus(*Quality*, *SubStatus*)

Changes the **Substatus** information (from 0 to 15) of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts. The OPC Standard specifies the following values:

- **Good Quality**:
  - **0**: non-specific
  - **1**: local override
- **Bad Quality**:
  - **0**: non-specific
  - **1**: configuration error
  - **2**: not connected
  - **3**: device failure
  - **4**: last known value
  - **5**: communication failure
  - **6**: out of service
- **Uncertain Quality**:
  - **0**: non-specific
  - **1**: last usable value
  - **4**: sensor not accurate
  - **5**: EU exceeded
  - **6**: subnormal

2.7.1.22 OPCSetVendor

OPCSetVendor(*Quality*, *Vendor*)

Changes the **Vendor Specific** information (from 0 to 255) of an OPC Quality and returns the modified value. This method is available in Links, as well as in scripts.

2.7.1.23 SetBit

SetBit(*Value*, *BitIndex*, *BitValue*)

Sets the value of *Value* (True or False) of the bit specified by *BitIndex* (from 0 to 31) to *BitValue*. This method returns an error in case the *BitIndex* parameter is outside the allowed range (between 0 and 31). This method is available in Links, as well as in scripts.

2.7.1.24 SourceTypeName

SourceTypeName(*SourceType*)

Returns a **String** with the description of the active Measurement Source (the **ActiveSource** property of Analog Measurement and Discrete Measurement objects in Elipse Power). This method is available in Links, as well as in scripts. Possible values for *SourceType* parameter are the following:

- **-1**: Empty **String**
- **0**: Active Source
- **1**: SCADA
- **2**: Operator
- **3**: Control Center
- **4**: Billing
- **5**: Calculated
- **6**: Database
- **100**: Topology Processor
- **101**: Power Flow
- **102**: State Estimator
- **103**: Load Shedding

NOTE: In case the value passed on *SourceType* parameter is not one of the possible values, this method returns "???".

2.8 General Events, Methods, and Properties of Objects

This section contains information about general **events**, **methods**, and **properties** of objects.

2.8.1 Events

Events are occurrences related to an object, which allow triggering scheduled actions. Basically, there are two types of events: **Physical** (or **External**) and **Internal**. **Physical Events** are, for example, user actions. In case users type on the keyboard, the relevant information can be the pressed key or, if users point and click the mouse, the relevant information can be the pointer position and the button status. **Internal Events** are, for example, value changes in a variable (Tag) in an application.

Since the Tag can be linked to an external device, it is possible to say that internal events can have physical associations, such as temperature changes in a chamber, for example.

2.8.1.1 Event Variables

Event Variables are created when the event starts. To be used, they must be linked to parameters in the event's script call.

The following example is a procedure call linked to the **KeyDown** event of `AnObject`.

```
Sub AnObject_KeyDown(KeyCode, Shift)
```

Notice that in this call there are two variables, *KeyCode* and *Shift*. E3 will automatically assign values to these variables when this event occurs. In this case, *KeyCode* will receive the code of the pressed key, and *Shift* will be True or False, whether the SHIFT key is pressed or not.

2.8.1.2 OnStartRunning

OnStartRunning()

Occurs as soon as an object starts. Example (`Months` is an Internal Tag and uses the **OnStartRunning** event to initialize an array):

```
Sub Months_OnStartRunning()
  Value = Array("January", "February", "March", "April",_
    "May", "June", "July", "August", "September", "October",_
    "November", "December")
End Sub
```

NOTE: To access this array, users must copy the **Value** property to a local variable.

2.8.1.3 OnStopRunning

OnStopRunning()

Occurs when the execution of an instance of this object finishes. Use the **OnStopRunning** event to perform termination operations for the object. Example:

```
Sub InternalTag1_OnStopRunning()
  ' When InternalTag1 object finishes
  ' it assigns False to InternalTag2
  Set tag2 = Application.GetObject("Data.InternalTag2")
  tag2.Value = False
End Sub
```

2.8.2 Methods

This topic lists common methods of E3 objects. Each entry displays the name of the method with its respective parameters, in its correct syntax, and also a usage example.

2.8.2.1 Calling Methods

Many predefined methods have parameters, which can (or should) be passed at method's call. For this, VBScript has a rule that must be followed. If the method is used in an assignment, its parameters must be enclosed in brackets. For example, check this **GetObject** method's call:

```
obj = Application.GetObject("data.tag001")
```

If a method is called alone, then users must remove the brackets. For example, check this **SetVariableValue** method's call:

```
Screen.Item("Query").SetVariableValue Value, 12
```

Brackets used when citing methods in this manual are only an indication to differentiate them from properties. In scripts, this rule must be followed.

2.8.2.2 Activate

Activate()

Activates a currently inactive object. Example:

```
Sub CommandButton1_Click()
  Dim obj, tag
  Set obj = Application.GetObject("Data")
  ' Create a new object and let it disabled (False).
  Set tag = obj.AddObject("DemoTag", False)
  ' Setup the new object parameters.
  tag.Name = "tag001"
  tag.Type = 3
  ' Enable the object (start it up).
  tag.Activate()
End Sub
```

2.8.2.3 AddObject

AddObject(*ClassName*[, *Activate*[, *ObjectName*]])

The **AddObject** method adds a new object in the application. This method has the *ClassName* parameter, which indicates the type of the object to create. For example, to create a rectangle on a Screen, the *ClassName* parameter must be "DrawRect". The created object is contained inside the object that called the **AddObject** method, and can be accessed using the **Item** method.

The *Activate* parameter is optional and indicates whether the object is enabled after creating it or not. If the object is activated, Links and scripts remain enabled. If the object is created with *Activate* equal to False, it can be later activated using the **Activate** method.

The *ObjectName* parameter is also optional and indicates a name for the new object. In case this name already exists, the new name is automatically incremented. If this parameter is not informed, the new object is named after the class defined in the *ClassName* parameter.

The object is created only if its type is compatible with the object that contains it. To ensure that the object was created, the **IsObject** method can be used.

NOTE: Only objects that have the **Insert** menu option can use this method.

2.8.2.4 Context

Context(*ContextName*)

Returns an object implementing the context indicated by the *ContextName* parameter, which must be a **String** surrounded by quotation marks. This method fails if no object on the upper hierarchy of the object calling this method implements this context. The following contexts are available:

- **Container**: Server and Viewer objects (objects inserted in project files or on folders inside projects)
- **Area**: Alarm Areas, or any server object whose property **IsAlarmArea** is set to True

NOTE: Context names are case-insensitive. To check the context to which an object belongs, open its Properties window, select the **Item** tab, and then check its valid contexts in the **Contexts** frame. In case an object defines more than one context, their names are presented in alphabetical order, separated by commas.

2.8.2.5 Deactivate

Deactivate()

Deactivates a previously created object, or an object previously activated using the **Activate** method. Users can deactivate an object when it is necessary to perform a previous configuration (property initialization, for example), or when they want to perform tests in which it is necessary that the object is neither present nor activated. Example:

```
Sub CommandButton1_Click()
  Dim obj, newObj
  Set obj = Application.GetObject("Data")
  Set newObj = obj.AddObject("DemoTag", True)
  ' Deactivate the object.
```

```
  newObj.Deactivate()
End Sub
```

2.8.2.6 DeleteObject

DeleteObject(*ChildName*)

Erases the specified object from the project. The *ChildName* parameter is a **String** (ignores capitalization) and indicates the child object to delete. This method returns True if the object was successfully deleted, or False, if the child object does not exist.

To delete an object using a reference to an element, the parent object's **DeleteObject** method is used. Example:

```
Sub CommandButton1_Click()
  Set obj = Application.GetObject("Data")
  If obj.DeleteObject("Tag001") Then
    MsgBox("Tag successfully deleted!")
  Else
    MsgBox("Deleting failed: tag does not exist.")
  End If
End Sub
```

NOTE: Only objects that have the **Insert** menu option can use this method.

2.8.2.7 GetChildObject

GetChildObject(*ObjectPath*)

The **GetChildObject** method returns a reference to the child object pointed by the *ObjectPath* parameter. Then it is possible to access all properties and methods of this object, similar to the **GetObject** method. This method fails if the path pointed by the *ObjectPath* parameter contains a property or method at its end. The path pointed by the child object is not a path starting at root (the .prj file) but instead a path starting at the object where this method is called.

NOTE: This method **DOES NOT** exist in the server's Application object nor in the Application Folders, however it exists in the Viewer's Application object, and it is accessible even in a **Read-Only** Viewer.

2.8.2.8 GetObject

GetObject(*ObjectPath*)

This method returns a reference to the object specified by *ObjectPath*. This allows accessing all object's properties and methods. This is a common practice in E3 scripts programming. It facilitates handling objects and makes code more intelligible. Example:

```
Sub CommandButton1_Click()
```

```
  ' Assign the value 20 to InternalTag1's Value property
  ' which is inside Data.
  Set tag = Application.GetObject("Data.InternalTag1")
  tag.Value = 20
End Sub
```

2.8.2.9 Item

Item(*ItemId*)

Returns a reference to the child object *ItemId* of the object that called this method. The **Item** method can search for an object either by its name or by its index (an integer starting at 1 and less or equal to the **Count** property). If the specified index or name is valid, the **Item** method returns the object's reference. Otherwise it will return an "Invalid Parameter" error. Example:

```
Sub Screen1_Click()
  ' Assigns to obj the reference to the child object Button1
  ' of Screen1.
  Set obj = Item("Button1")
  ' Set the obj's BackColor property, that is,
  ' Button1.
  obj.BackColor = RGB(255, 0, 0)
End Sub
```

2.8.2.10 Save

Save()

This method saves the specified object, which was modified at run time. Children objects will also be saved, according to the parent object's specifications. This method is not valid for Screen and Viewer objects. Example:

```
Sub CommandButton1_Click()
  Set area = Application.GetObject("ConfigAlarms")._
    AddObject("Area", true)
  Application.GetObject("ConfigAlarms").Save()
End Sub
```

NOTE: Changes made at run time and saved in the object are only visible in Studio after updating the project, which can be performed by right-clicking the project's name and then selecting the **Refresh** option.

2.8.3 Properties

Each object has **Properties**, which are used to save information about its characteristics. For example, a **Rectangle**-type object has a **Name** property, which stores the object's name, and the **Width** and **Height** properties, which store its width and height, respectively, among other properties.

This chapter lists all properties common to E3 objects. Each entry has the name of

the property, its description and, when applicable, an example of its usage.

Properties are identified by an icon indicating the data type supported by their content. The available data types are the following:

Available data types

ICON	DATA TYPE	DESCRIPTION
	Boolean	Returns True or False.
	Numeric	Returns an integer or double (positive or negative), as defined by the property.
	Date	Returns a date in the Gregorian calendar (beginning in 1899).
	Text	Returns a text.
	Variant	Returns a variable type, which can assume several formats.
	Color	Returns a color in RGB format.
	Link	Returns a Link between objects.
	Enum	Returns a determined set of values.

Some properties can propagate their values to the same property in their child objects. In this case, they are called propagatable properties. Users can, however, force a child object's property to behave distinctively.

NOTE: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is equivalent to a thousandth centimeter; that is, 1,000 units is equivalent to one centimeter. Thus, this is the standard adopted when describing E3 properties, when applicable.

2.8.3.1 Application

The **Application** property returns the Application object related to the current object's context. With this Application object it is possible, for example, to search for other objects in the application. Example:

```
Sub Screen1_Click()
  ' When clicking the screen, sets a value
  ' and displays a message box
  set obj = Application.GetObject("Data.InternalTag1")
  obj.Value = 100
  MsgBox "Value of InternalTag1: " & obj.Value
```

```
End Sub
```

2.8.3.2 Count

9 Returns the number of child objects(items) an object contains. This property works together with the **Item** method. If the object does not have any child objects, the value of this property is 0 (zero). Example:

```
Sub Screen1_Click()
  ' Searches all Screen objects and sets
  ' the ForegroundColor property to red
  Dim obj
  For i = 1 To Count
    Set obj = Item(i) ' Gets a child object
    obj.ForegroundColor = RGB(255, 0, 0)
  Next
End Sub
```

2.8.3.3 DocString

A Free text that enables documenting object's functionality or features by the project's programmers. Example:

```
Sub CommandButton1_Click()
  Docstring = "This button activates the system condenser."
  MsgBox Docstring
End Sub
```

2.8.3.4 Links

Returns an object that is a collection of connections (or Links) of any E3 object. This property is only accessible at run time. Check the item **Collection of Links** for more information about the collection of objects returned by this property.

2.8.3.5 Name

A Identifies every object present in the application. Modifying this property implies in modifying all other properties or scripts that use this object. Hence, it is not advisable to change this property at run time. Example:

```
Sub CommandButton9_Click()
  MsgBox "Screen's name is " & (Screen.Name)
End Sub
```

2.8.3.6 Parent

Returns the object's parent object. So, if an object is inserted on a Screen, then the **Parent** property returns "Screen". Likewise, if an Internal Tag is inserted directly under a Data Server, InternalTag's **Parent** property then points to the Data Server. Example:

```
Sub Rectangle1_Click()
```

```
  ' When clicking Rectangle1,
  ' changes the foreground color of Rectangle2
  Parent.Item("Rectangle2").ForegroundColor = RGB(255, 0, 0)
End Sub
```

2.8.3.7 PathContainer

A Returns a **String** with the path of the object containing the current object, including its Folders. This value is determined only at request, thus it is not recommended to create Links to this property.

2.8.3.8 PathName

A Identifies a path name in the application. This property is only available at run time. Example:

```
Sub CommandButton9_Click()
  MsgBox "Screen path is " & (Screen.PathName)
End Sub
```

2.8.3.9 PathVolume

A Returns a **String** with the name of the .prj or .lib file containing the object. In Studio, this property returns the project's or library's full path (c:\folder\folder \volume.prj). At run time, the objects running in the Viewer always return an empty **String**. However, objects running in the server return the project's or library's relative path, according to the way it is stored in the Domain (volume.prj). This value is determined only at request, thus it is not recommended to create Links to this property.

2.8.4 Collection of Links

This section contains information about **methods** and **properties** common to the collection of connections (or Links) of any E3 object, returned by the **Links** property.

2.8.4.1 Common Methods

This section contains information about methods common to the collection of Links returned by the **Links** property.

2.8.4.1.1 CreateLink

CreateLink(*Property*, *Source*[, *Type*])

This method allows creating a Link with an object's property. In case of success, this method returns the object just created. Otherwise, a script error occurs and this method returns `Nothing`. This method has the following parameters:

- ***Property***: specifies the name of the property to which the Link will be created

- ***Source***: specifies the name of Link's source object
- ***Type*** (optional): specifies the type of Link to create. When omitted, creates a Simple Link

NOTE: Not all existing properties in an object allow creating Links. To check which properties allow this feature, go to the **Links** tab. If the property is invalid for a Link, it does not exist, or it already has a Link, then a script error occurs.

Available options for Type parameter

OPTION	DESCRIPTION
0 - Simple Link	In a Simple Link, its source value is copied to the property every time it is modified.
1 - Bidirectional Link	A Bidirectional Link works as a Simple Link, but when there is a variation in the property, its value is copied to the source, thus generating a two-direction Link.
2 - Analog Link	An Analog Link specifies a conversion scale between the source variable and the property.
3 - Digital Link	In a Digital Link, fixed or alternating values are set to the property, and these values are set according to whether the source is True or False.
4 - Table Link	In a Table Link, users can specify conditions between the variables, the values, and the destination. On the table users can specify minimum and maximum values, and other configurations.
5 - Reverse Link	A Reverse Link is a unidirectional Link from the property to the source.
6 - Multisource Link	A Multisource Link is similar to a Table Link, except that each Link row allows retrieving its value from a different source.

Example:

```
Sub CommandButton1_Click()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Text1").Links.Item("Value")
  If Bind Is Nothing Then
    MsgBox "Text1 is not linked to any object."
```

```
    Dim Source
    Source = "Data.InternalTag1.Value"
    MsgBox "Creating a link in '" & Source & "'."
    Set Bind = Screen.Item("Text1").Links._
      CreateLink("Value", Source, 0)
    Bind.BiDirectional = Screen.Item("BiDirectional").Value
    Bind.Reverse = Screen.Item("Reverse").Value
    MsgBox "Type: " & TypeName(Bind)
  Else
    MsgBox "Text1 is already linked to '" &Bind.Source& "'."
  End If
End Sub
```

2.8.4.1.2 Item

Item(*Property*, *Index*)

This method returns a Link object from a specific property of an object. If it is a text, *Property* specifies the name of the property to access. The Link can also be numerically accessed by *Index*. This index must be inside the range 1 up to **Count**. In case there is no Link to the property, or the index is invalid, then a script error occurs. As in other collections, The Links collection allow using VBScript's **For Each** command. Example:

```
Sub Texto1_Click()
  For Each Link In Links
    MsgBox "Link source: " & Link.Source
  Next
End Sub
```

2.8.4.1.3 RemoveLink

RemoveLink(*Property*)

This method removes a Link from a property specified by *Property*, if it exists. If a Link to the specified property does not exist, this method has no effect. Example:

```
Sub CommandButton2_Click()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  If Bind Is Nothing Then
    ' If the link does not exist
    MsgBox "ScrollBar1 is not linked."
  Else
    MsgBox "ScrollBar1 is linked to '" & Bind.Source & "'"
    MsgBox "Removing the link."
    Screen.Item("ScrollBar1").Links.RemoveLink("Value")
  End If
End Sub
```

2.8.4.2 Common Properties

This section contains information about properties common to the collection of Links returned by the **Links** property.

2.8.4.2.1 Count

9 Returns the number of child objects (items) of a Link collection. This property works together with the **Item** method. If the collection does not have children, the returned value is 0 (zero).

2.8.4.3 Links

This section contains information about **Link**-type objects inside the collection of Links returned by the **Links** property. The available types of Links are the following:

- **Simple**
- **Bidirectional**
- **Analog**
- **Digital**
- **Table**
- **Reverse**
- **Multisource**

2.8.4.3.1 Common Properties

This section contains information about the properties common to objects contained in the collection of Links returned by the **Links** property.

2.8.4.3.1.1 Property

A Specifies the name of the property it is linked to. When modified, moves the Link to another property of the same object. Example:

```
Sub CommandButton1_Click()
  Dim bind
  Set bind = Screen.Item("TableBind").Links.Item(1)
  bind.Property = "Caption"
End Sub
```

2.8.4.3.1.2 Source

A Specifies Link's source, which can be the name of another application object, or a more complex expression accessing several objects. Example:

```
Sub CommandButton25_Click()
  Dim bind
  Set bind = Screen.Item("TableBind").Links.Item(1)
  bind.Source = "Data.DemoTag1.Value"
End Sub
```

2.8.4.3.1.3 Type

This is a read-only property, and informs the Link's type. The available options are:

Available options for Type

OPTION	DESCRIPTION
0 - bsSimple	Simple Link.
1 - bsSimpleBiDir	Bidirectional Link.
2 - bsAnalog	Analog Link.
3 - bsAnimation	Digital Link.
4 - bsTable	Table Link.
5 - bsReverse	Reverse Link.
6 - bsMultiSource	Multisource Link.

2.8.4.3.2 Simple Link

The Simple Link object does not have events, methods, or properties associated to it.

2.8.4.3.3 Bidirectional Link

This section contains information about **properties** of a Bidirectional Link object. This object does not have associated events nor methods.

2.8.4.3.3.1 Properties

This section contains information about properties of a Bidirectional Link object.

BiDirectional

True when a Link is Bidirectional, False when a Link is Simple or Reverse.

2.8.4.3.4 Analog Link

This section contains information about **properties** of an Analog Link object. This object does not have associated events nor methods.

2.8.4.3.4.1 Properties

This section contains information about properties of an Analog Link object.

DstHiValue

9 Specifies the highest value reached by the property. Example:

```
Sub DstHiValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar1").Max = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing DstHiValue from " &_
      Bind.DstHiValue & " to " & Value
    Bind.DstHiValue = Value
  End If
End Sub
```

DstLoValue

9 Specifies the lowest value reached by the property. Example:

```
Sub DstLoValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar1").Min = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing DstLoValue from " &_
      Bind.DstLoValue & " to " & Value
    Bind.DstLoValue = Value
  End If
End Sub
```

SrcHiValue

9 Specifies the highest value reached by the source. Example:

```
Sub SrcHiValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar2").Max = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing SrcHiValue from " &_
      Bind.SrcHiValue & " to " & Value
    Bind.SrcHiValue = Value
```

```
  End If
End Sub
```

SrcLoValue

9 Specifies the lowest value reached by the source. Example:

```
Sub SrcLoValue_ValueChange()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("ScrollBar1").Links.Item("Value")
  Screen.Item("ScrollBar2").Min = Value
  If Bind Is Nothing Then
    MsgBox "ScrollBar1 has no link."
  Else
    MsgBox "ScrollBar1 is linked to " & Bind.Source & "'"
    MsgBox "Changing SrcLoValue from " &_
      Bind.SrcLoValue & " to " & Value
    Bind.SrcLoValue = Value
  End If
End Sub
```

NOTE: In case the values specified for **SrcHiValue** and **SrcLoValue** properties are the same, there will be no scale, then the Link will work as a Simple Link.

2.8.4.3.5 Digital Link

This section contains information about **properties** of a Digital Link object. This object does not have associated events nor methods.

2.8.4.3.5.1 Properties

This section contains information about properties of a Digital Link object.

BlinkOff

◩ When this property is set to True, the connected property alternates periodically between values of **OffValue** and **BlinkOffValue** properties, in case the source returns False. Example:

```
Sub BlinkOff_Change()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Rectangle1")._
    Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
    MsgBox "Changing BlinkOff from " &_
      Bind.BlinkOff & " to " & Value
    Bind.BlinkOff = Value
```

```
  End If
End Sub
```

BlinkOffValue

Specifies an alternative value the property assumes periodically when the source's expression returns False, and the **BlinkOff** property is set to True. Example:

```
Sub BlinkOffValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & "'"
      MsgBox "Changing BlinkOffValue from " &_
        Bind.BlinkOffValue & " to " & Value
      Bind.BlinkOffValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

BlinkOn

When this property is set to True, the connected property alternates periodically between values of **OnValue** and **BlinkOnValue** properties, in case the source returns True. Example:

```
Sub BlinkOn_Change()
  On Error Resume Next
  Dim Bind
  Set Bind = Screen.Item("Rectangle1")._
    Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
    MsgBox "Changing BlinkOn from " &_
      Bind.BlinkOn & " to "_ & Value
    Bind.BlinkOn = Value
  End If
End Sub
```

BlinkOnValue

Specifies an alternative value the property assumes periodically when the source's expression returns True, and the **BlinkOn** property is set to True. Example:

```
Sub BlinkOnValue_Click()
```

```
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to " & Bind.Source & "'"
      MsgBox "Changing BlinkOnValue from " &_
        Bind.BlinkOnValue & " to " & Value
      Bind.BlinkOnValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

OffValue

Specifies the value a property assumes when the source's expression returns False. Example:

```
Sub OffValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & "'"
      MsgBox "Changing OffValue from " &_
        Bind.OffValue & " to " & Value
      Bind.OffValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

OnValue

Specifies the value a property assumes when the source's expression returns True. Example:

```
Sub OnValue_Click()
  On Error Resume Next
  Dim Value
  If Application.ShowPickColor_
    (Value, ForegroundColor, 400, 300) Then
```

```
    Dim Bind
    Set Bind = Screen.Item("Rectangle1").Links._
      Item("ForegroundColor")
    If Bind Is Nothing Then
      MsgBox "Rectangle1 has no link."
    Else
      MsgBox "Rectangle1 is linked to "& Bind.Source & " "
      MsgBox "Changing OnValue from " &_
        Bind.OnValue & " to " & Value
      Bind.OnValue = Value
    End If
    ForegroundColor = Value
  End If
End Sub
```

2.8.4.3.6 Table Link

This section contains information about **methods** and **properties** of a Table Link object. This object does not have associated events.

2.8.4.3.6.1 Methods

This section contains information about methods of a Table Link object.

InsertRow

InsertRow([*Row*])

Inserts a new row on the table. The *Row* parameter is optional and specifies at which table position the row should be inserted. When omitted, it assumes the standard behavior of inserting a row at the end of the table, which is the same as using *Row* equal to -1. When it is informed and is not equal to -1, it must be a value between 1 and **Count**, and the new row moves the other rows greater or equal to the index towards index ascending direction. A new row always assumes the following standard values for its properties:

- **Min**: 0.0
- **Max**: 1.0
- **Blink**: False
- **BlinkValue**: 0.0
- **Value**: 0.0

Example:

```
Sub Rectangle1_Click()
  On Error Resume Next
  Dim Bind
  Set Bind =
```

```
Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no links."
  Else
    Dim row
    row = Screen.Item("SetPointRow").Value
    MsgBox Bind.RowCount
    If (row < 1 Or row > Bind.RowCount) Then
      MsgBox "Invalid row number: " & row
    Else
      MsgBox "Adding a row at: " & row
      Bind.InsertRow(row)
      If row = -1 Then
        row = Bind.RowCount
        Bind.Value(line) =
Screen.Item("RectangleValue").ForegroundColor
        Bind.BlinkValue(line) =
Screen.Item("RectangleBlinkValue").ForegroundColor
        Bind.Max(line) = Screen.Item("SetPointMax").Value
        Bind.Min(line) = Screen.Item("SetPointMin").Value
        Bind.Blink(line) = Screen.Item("CheckBoxBlink").Value
      End If
    End If
  End If
End Sub
```

Item

Item(*ItemId*)

Returns a reference to a Table Link Row object, indicated by *ItemId*.

RemoveRow

RemoveRow(*Row*)

Removes a row at a specified index. The *Row* parameter determines the table row to remove (it must be a value between 1 and **Count**). Example:

```
Sub RemoveRow_Click()
  On Error Resume Next
  Dim Bind
  Set Bind =_
    Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  If Bind Is Nothing Then
    MsgBox "Rectangle1 has no link."
  Else
    MsgBox "Rectangle1 is linked to '" & Bind.Source & "'"
    Dim row
    row = Screen.Item("Row").Value
    MsgBox "Removing line " & row
    Bind.RemoveRow row
  End If
```

```
End Sub
```

2.8.4.3.6.2 Properties

This section contains information about properties of a Table Link object.

Count

9 Informs the number of rows on a table. This is a read-only property.

2.8.4.3.6.3 Table Link Row

This section contains information about **properties** of a Table Link Row object. This object does not have associated events nor methods.

Propriedades

This section contains information about properties of a Table Link Row object.

Blink

◩ Determines that when the source is within this Row's interval, the property alternates periodically between values established in the **Value** and **BlinkValue** properties. Example:

```
Sub CheckBox1_Click()
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).Blink = Value
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(2).Blink = Value
End Sub
```

BlinkValue

v Specifies the alternative (blinking) value the property assumes when source is within the specified Row's interval, and the **Blink** property is set to True. Example:

```
Sub CommandButton1_Click()
  Dim Color
  ' Picks a color
  Application.ShowPickColor Color, 0, 100, 100
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).BlinkValue = Color
End Sub
```

Max

9 Specifies the highest source value for a Table Row.

Min

9 Specifies the lowest source value for a Table Row. Example (applicable for both **Max** and **Min** properties):

```
Sub CommandButton1_Click()
  Set Bind = _
```

```
    Screen.Item("Rectangle1").Links.Item("ForegroundColor")
  Set Line1 = Bind.Item(1)
  Line1.Min = 0
  Line1.Max = 20
  Set Line2 = Bind.Item(2)
  Line2.Min = 21
  Line2.Max = 100
End Sub
```

Value

Specifies the value the property assumes when the source is within the specified Row's interval. Example:

```
Sub CommandButton1_Click()
  Dim Color
  ' Picks a color
  Application.ShowPickColor Color, 0, 100, 100
  Screen.Item("Rectangle1").Links.Item("ForegroundColor")._
    Item(1).Value = Color
End Sub
```

2.8.4.3.7 Reverse Link

This section contains information about **properties** of a Reverse Link object. This object does not have associated events nor methods.

2.8.4.3.7.1 Properties

This section contains information about properties of a Reverse Link object.

Reverse

True when a Link is Reverse, False when a Link is Bidirectional or Simple.

2.8.4.3.8 Multisource Link

This section contains information about **methods** and **properties** of a Multisource Link object. This object does not have associated events.

2.8.4.3.8.1 Methods

This section contains information about methods of a Multisource Link object.

InsertRow

InsertRow(*InsertAtRow*)

Inserts a new Row on the Multisource Link table.

Item

Item(*ItemId*)

Returns a reference to a Multisource Link Row object, indicated by *ItemId*.

RemoveRow

RemoveRow(*Row*)

Removes the Row at the index indicated by the *Row* parameter.

2.8.4.3.8.2 Properties

This section contains information about properties of a Multisource Link object.

AdviseAll

◩ This property keeps all Links of the Multisource Link table in **Advise** (active) mode. The default value of this property is True. For applications created in previous versions, this property is set to False, for compatibility reasons.

Count

9 This property returns the number of Rows on the Multisource Link table.

2.8.4.3.8.3 Multisource Link Row

This section contains information about **properties** of a Multisource Link Row object. This object does not have associated events nor methods.

Propriedades

This section contains information about properties of a Multisource Link Row object.

Max

9 Maximum interval for a Link value indicated by the **Source** property.

Min

9 Minimum interval for a Link value indicated by the **Source** property.

Source

A Specifies Multisource Link Row's source, which can be the name of another application object, or a more complex expression accessing several objects.

CHAPTER 3

User Libraries

This section contains information about XControls and XObjects, and also ElipseX Properties.

3.1 XControls and XObjects

This section contains information about **events** and **properties** of XControls and XObjects. These objects do not have methods associated to them.

3.1.1 Events

This section contains information about the events of XControls and XObjects.

3.1.1.1 Constructor

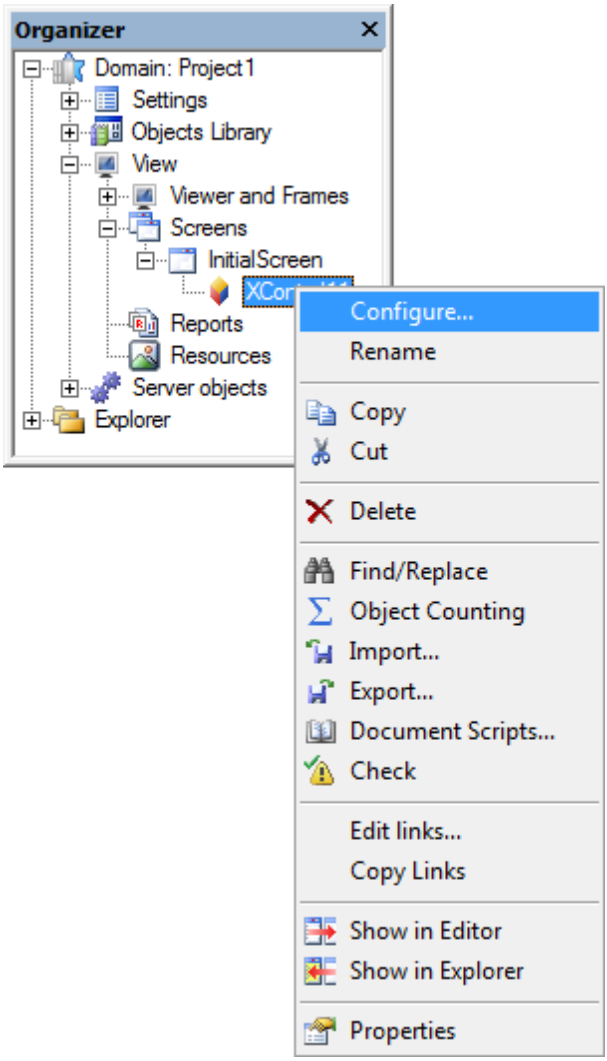
Constructor()

Triggered when the ElipseX is started. Users can use this event to run a script that sets internal values of an ElipseX, for example.

3.1.1.2 CustomConfig

CustomConfig()

Allows automated configurations on ElipseX instances. A configuration option appears on a contextual menu of an ElipseX instance when there is a script linked to the **CustomConfig** event of the ElipseX definition. When this option is selected on this menu, that event is triggered. The text that appears on this menu option can be set on the **CustomConfigText** property of the ElipseX definition. If this property is empty, the displayed text is "Configure".



Contextual menu of an ElipseX instance

NOTE: A script linked to the **CustomConfig** event runs on Studio, where objects are not active. Therefore, its behavior is different from the usual.

3.1.2 Properties

This section contains information about the properties of XControls and XObjects.

3.1.2.1 CustomConfigText

A Indicates a text that appears on the contextual menu for the configuration option of the ElipseX instance. This option only appears if there is a script associated to the **CustomConfig** event of the ElipseX definition. If the property value is empty, the text "Configure" is then displayed in the menu option. The default value of this property is is an empty **String**.

3.2 ElipseX Properties

This section contains information about **events** and **properties** of the Property object of an ElipseX. This object does not have methods associated to it.

3.2.1 Events

This section contains information about the events of the Property object of an ElipseX.

3.2.1.1 OnPropertyChanged

OnPropertyChanged()

It is triggered when an ElipseX property is modified. You may use this event to trigger scripts that execute actions according to a certain ElipseX status.

3.2.2 Properties

This section contains information about the properties of the Property object of an ElipseX.

3.2.2.1 DefaultValue

V Defines the Property's initial value when a new instance of the object is created. If the Property is configured as **retentive**, then it is initialized with this value every time the object is loaded. The default value of this property is Empty.

3.2.2.2 HelpString

A Text containing the Property's description. This text is displayed at the bottom of the Property List in Studio when the Property is selected. The default value of this property is an empty **String**.

3.2.2.3 Persistable

◩ Indicates whether the Property is saved in the project file (True) or is available only at run time (False). The default value of this property is True. If this property is

configured as False, the Property cannot be edited in Studio anymore, saved nor read from the project file. However, the Property is still visible in the AppBrowser. This property is represented by the icon .

When this property is configured as True, the Property receives its default value (the **DefaultValue** property) only when creating an instance. If the **DefaultValue** property changes, the object instances already created are not affected.

When this property is configured as False, the Property receives its default value whenever an instance is loaded, that is, whenever the **DefaultValue** property changes, all instances already created are initialized with the new default value.

3.2.2.4 Public

When an ElipseX Property is public (True), it is visible outside the Library. Otherwise, the Property is internal and only visible to the object. The default value of this property is True. A public Property is represented by the icon .

3.2.2.5 Retentive

Indicates if the current Property's value at run time is persisted in the Domain file (True), while the Domain is loaded. Retentive Properties have the following behavior:

- Propagates its value to the server in Standby
- Keeps its value if the application is updated at run time
- Keeps its value if the application is stopped (as long as the Domain is not unloaded)

NOTES:
- This property is only available for Properties whose data type is not an object (**Variant**, **Double**, **Integer**, etc.)
- Using this property in True implies in raising E3Run's memory and CPU usage, therefore it must be used carefully

3.2.2.6 Type

A Determines the type of values accepted by a Property (for example, **Boolean**, **Double**, **Integer**, **Variant**, etc.). When specifying an object type (as for example DemoTag, IOTag, XObject, etc.), this property has the following behavior:

- **In case the ElipseX is inactive**: This property works as a **String**, which specifies the object's instance path of the configured type
- **In case the ElipseX is active**: When writing, this property works just like when the object is inactive. However, when reading this property returns the specified object, in case it exists. If the path does not point to an existing object at the time,

this property returns `Nothing`

CHAPTER 4

View

This section contains information about events, methods, and properties of E3's view objects:

- **Viewer**
- **Frames and Splitters**
- **Screen and Screen Objects**
- **E3Alarm**
- **E3Browser**
- **E3Chart**
- **E3Playback**
- **Reports**

4.1 Viewer

This section contains information about **events**, **methods**, and **properties** of the Viewer object.

4.1.1 Events

This section contains information about the events of the Viewer object.

4.1.1.1 OnInactive

OnInactive()

This event occurs while Viewer is inactive, if the **EnableInactivity** property is set to True. It is triggered when determining that a user is not using Viewer for a period of time equal or greater than the value of the **InactivityTime** property.

In a script for this event, users may program what to do when Viewer is inactive for a certain period of time. For example, it is possible to perform a Viewer *logout* after 20 minutes without activity. Example:

```
Sub Viewer_OnInactive()
  Logout(false)
  If MsgBox("This Viewer section was closed
    due to inactivity.") = 0 Then
    Application.GetFrame("").OpenScreen "InitialScreen", 0
End Sub
```

4.1.1.2 User Events

This section contains information about user events of the Viewer object.

4.1.1.2.1 OnLogin

OnLogin()

Occurs when a user performs a successful application *login* (user authentication). An application *login* can be performed using the **Login** method or when an object that can only be accessed by users with a certain authorization level requests authentication.

4.1.1.2.2 OnLogout

OnLogout()

Occurs when performing a *logout*, that is, a user quits an application. A *logout* is performed by calling the **Logout** method.

4.1.2 Methods

This section contains information about the methods of the Viewer object.

4.1.2.1 CaptureScreen

CaptureScreen(*Filename*)

The **CaptureScreen** method captures the current Screen and saves it to a file. This methods has the same effect of the PRINT SCREEN key on Windows. The *Filename* parameter determines a file name to which this Screen is saved.

NOTE: The final file format is Bitmap (.bmp), even if the *Filename* parameter informs another file extension (.gif, .jpg, etc.).

Example:

```
Sub CommandButton1_Click()
  Screen.Frame.CaptureScreen "c:\temp\screen.bmp"
End Sub
```

4.1.2.2 ChangePassword

ChangePassword()

This method opens a dialog box to change current user's password. This method returns True if the current user has permission to change their password. Otherwise, it returns False, indicating an operation failure, or that it is not possible

to change that password, because this user is not authorized.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.3 DoModal

DoModal(*Screen*, *Title*, *Left*, *Top*, *Width*, *Height*, *Arg*, *Flags*)

Opens a modal Screen, which is a window that do not allow clicking other Screens or windows while it is not closed. The *title* parameter passed in this method is only used if the **Caption** property is empty. Otherwise, this parameter is ignored. This method has the following parameters:

- **Screen**: Determines a Screen's name
- **Title**: Determines a title for this modal window
- **Left, Top**: XY position of this modal window, in pixels
- **Width**: Width of this modal window, in pixels or Himetric
- **Height**: Height of this modal window in pixels or Himetric
- **Arg**: Determines a variable to use on Screen's **OnPreShow** event
- **Flags**: Determines a combination used on this modal window. Such combination is performed by summing values of the next table that corresponds to user options. When the specified value is equal to -1 (minus one), Viewer configurations are set to this modal window. When this value is different from -1 (minus one), users can use all combinations described on the next table

Possible combinations for Flags parameter

VALUE	DESCRIPTION
1	Enables a title bar on this window
2	Enables a **Close** button on this window
4	Enables a **Minimize** button on this window
8	Enables a **Maximize** button on this window
16	Enables a window border
32	Specifies that this window can be resized
64	Specifies that this window can be moved
256	Specifies that this window stays on top of a Screen
512	Specifies that this window is configured with a **Tool Bar** style
1024	Disables object buttons
2048	Centers this modal Screen on a Frame horizontally, as well as vertically

Example:

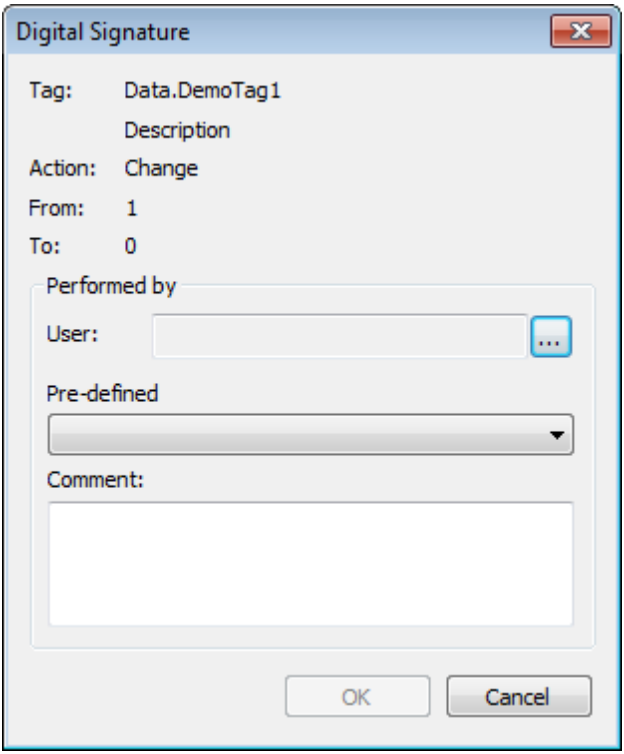
```
Sub Button1_Click()
  ' When clicking the Button1 opens another modal Screen
  Application.DoModal "Screen1", "Title", 0, 0, 400, 200, 0, 3
End Sub
```

NOTE: Size values on this method can be informed as numbers or **Strings**. In case of numbers, they are interpreted as pixels. In case of **Strings**, if followed by an "hm" unit, they are interpreted as Himetric. Any other case is considered as pixels.

4.1.2.4 ESign

ESign(*ObjName*[, *Description*[, *Action*[, *From*[, *To*[, *User*[, *Comment*]]]]]])

The **ESign** method is used to validate a field change, using an electronic signature. When this method is used, the dialog box on the next figure is displayed.

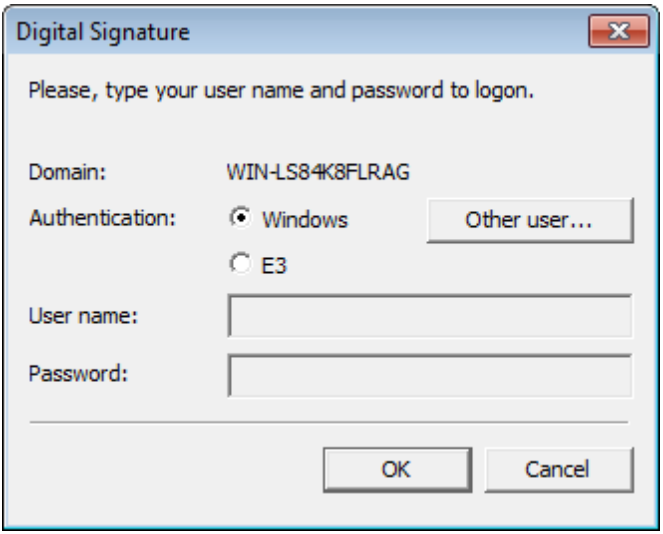


Digital Signature dialog box

Parameters of the ESign method

PARAMETER	DESCRIPTION
ObjName	Text that contains a Tag name or another application object.
Description	Text that contains a description for **ObjName**. This parameter is optional and, if omitted, this dialog box tries to retrieve data from **ObjName**'s **DocString** property.
Action	Text that contains an action to execute (for example, "Value change"). This parameter is optional and its default value is an empty **String**.
From	**Variant** that contains the original value, or a state to alter. This parameter is optional.
To	**Variant** that contains the new Tag value, or a value to apply on **Action**. This parameter is optional.
User	Returned text. Receives the *login* name selected on this dialog box. This parameter is optional.
Comment	Returned text. Receives a comment typed on this dialog box. This parameter is optional.

When clicking , a window then opens to authenticate a user. If the **Windows** option is selected, the **User name** and **Password** fields are automatically disabled. Click **Other user** to select a user belonging to a network domain. In case the **E3** option is selected, type information about a user belonging to an E3 Domain in the **User name** and **Password** fields.



Integrated login

This method returns True if users click **OK**, and if the **User** and **Password** fields are valid. Otherwise, if this dialog box is dismissed or if the *login* or password are wrong after three attempts, this method returns False. In case of failure, **User** and **Comment** are configured to an empty **String**.

Pre-defined comments are stored on Windows Registry. Only the last 26 comments are saved. Every time this window is created, the last comments are retrieved from Registry, and then used to populate a list box. If users provide a new comment, it is saved and the oldest one is discarded, in case there is no free position. If it is an already existing comment, then it moves to the top of the recent comments list. Example:

```
Sub Button1_Click()
  Dim Tag, User, Comment
  Set Tag = Application.GetObject("IO.Inputs.IO01")
  If Application.ESign(Tag.PathName, , "Value Change", _
    Tag.Value, 1, User, Comment) Then
    If Tag.WriteEx = 1 Then
      Application.TrackEvent _
        "Tag IO.Inputs.IO01 changed to 1 " &_
        "by the user" & User, Comment
    End If
  End If
End Sub
```

4.1.2.5 ExecuteExternalApp

ExecuteExternalApp(*AppPath*, *Arguments*, *InitialDir*, *CmdShow*[, *ProcessId*])

This method executes an external application with *AppPath* name and path, with

Arguments, and starting at the *InitialDir* working directory. When a document is specified in *AppPath*, the application associated to this document is executed, and this document is passed as one of the parameters of that application. *ProcessID* receives a number that identifies a process (this number is used in the **IsAppRunning** method and is the same value that appears on Windows Task Manager, on **PID** column). The *CmdShow* parameter specifies the opening mode of this application's window, as described on the next table.

Available options for CmdShow parameter

OPTION	DESCRIPTION
0	Hides this window and activates another window.
1	Activates and displays this window. If this window is maximized or minimized, it is restored to its original size and position. An application must specify this value when it is displaying a window for the first time.
2	Activates this window and displays it minimized.
3	Activates this window and displays it maximized.
4	Displays this window with its most recent size and position. The active window remains active.
5	Activates this window and display it with its current size and position.
6	Minimizes this window and activates the next higher level window.
7	Displays this window minimized. The active window remains active.
8	Displays this window in its current state. The active window remains active.
9	Activates and displays this window. If this window is maximized or minimized, it is restored to its original size and position. An application must specify this value when restoring a window that was minimized.

Example:

```
Sub CommandButton1_Click()
  Dim ret
  Application.ExecuteExternalApp "calc.exe", "", "", 1, ret
  Application.GetObject("Data.InternalTag1").Value = ret
End Sub
```

NOTE: The parameter returned by *ProcessID* can be 0 (zero), in case no process has been started. For example, if an open document is an URL and an Internet Explorer's instance is already running, it displays that document. No new process is started, thus *ProcessID* is 0 (zero).

4.1.2.6 Exit

Exit()

This method closes this window in Viewer. Example:

```
Sub_Button1.Click()
  Application.Exit()
EndSub
```

4.1.2.7 GetFormulaUnitDataObj

GetFormulaUnitDataObj(*FormulaName*)

This method gets all settings of the existing Units on a certain Formula. These Units are the destination of saved data in a Formula (values). This method has the *FormulaName* parameter, which informs Formula's name.

Use the **GetFormulaUnitDataObj** method to get a collection of Units of a Formula. This method returns True if this operation is successful or False otherwise. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a
  ' message box with the number of Units and the name
  ' of the First Unit
  Set obj = Application.GetFormulaUnitDataObj("Formula1")
  MsgBox CStr(obj.Count)
  MsgBox CStr(obj.Name(1))
End Sub
```

4.1.2.8 GetFormulaValueDataObj

GetFormulaValueDataObj(*FormulaName*)

This method gets all settings of the existing values of a certain Formula. Values are a data set saved in a Formula. This method has the *FormulaName* parameter, which informs Formula's name.

Use the **GetFormulaValueDataObj** method to get a collection of values in a Formula. This method returns True if this operation is successful or False otherwise. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a message box
  ' with the number of Sets
  ' and the name of the First Set.
  Set obj = Application.GetFormulaValueDataObj("Formula1")
  MsgBox CStr(Obj.Count)
```

```
  MsgBox CStr(obj.Name(1))
End Sub
```

4.1.2.9 GetFrame

GetFrame([*FrameName*])

This method searches for a Splitter object that is already open in the current Viewer. This method has a *FrameName* parameter, which is optional and determines a Frame's name to search for. If the value specified in *FrameName* is empty, it returns a Frame that contains all Splitters or the active Screen. With this method's return value, use Splitter methods, for example the **OpenScreen** method, to open another Screen. Example:

```
Sub Button1_Click()
  ' When clicking this Button, gets the 'Menu' Frame
  ' and replaces the current Screen of this Frame by the
'Options' Screen
  Set newFrame = Application.GetFrame("Menu")
  ' newFrame has a Splitter-type object
  newFrame.OpenScreen "Options", 0
End Sub
```

4.1.2.10 GetFullUserName

GetFullUserName()

The **GetFullUserName** method returns the complete name of a user logged in E3. In case there is no logged-in user, an empty **String** is then returned.

4.1.2.11 GetKeyPad

GetKeyPad()

Returns a reference to an Elipse KeyPad object, allowing to manipulate a floating virtual keyboard on applications developed using E3. Please check the **E3 User's Manual** for more information on this object. Methods and properties of Elipse KeyPad are described on chapter **ActiveX - Elipse KeyPad**.

4.1.2.12 GetMouseX

GetMouseX()

Returns mouse pointer's X coordinate, in pixels, relative to the entire area of computer's screen.

NOTE: This method fails in a script if the current mouse pointer position cannot be retrieved. One of the situations where this failure can be verified is when Windows Logon window opens (when pressing the CTRL + ALT + DEL key combination). This behavior of preventing access to the current mouse pointer position is default on Windows in some situations and cannot be overridden. As a suggestion, use the **On Error Resume Next** command before using this method to prevent script errors.

4.1.2.13 GetMouseY

GetMouseY()

Returns mouse pointer's Y coordinate, in pixels, relative to the entire area of computer's screen.

NOTE: This method fails in a script if the current mouse pointer position cannot be retrieved. One of the situations where this failure can be verified is when Windows Logon window opens (when pressing the CTRL + ALT + DEL key combination). This behavior of preventing access to the current mouse pointer position is default on Windows in some situations and cannot be overridden. As a suggestion, use the **On Error Resume Next** command before using this method to prevent script errors.

4.1.2.14 GetScreen

GetScreen(*ScreenName*)

This method returns a Screen object, whose name must be specified in the *ScreenName* parameter. The returned value of this method can be used on Splitter's **ShowScreen** method.

4.1.2.15 GetValue

GetValue(*TagName*)

The **GetValue** method searches for an object's value specified in the *TagName* parameter. If *TagName* points to a property, this method returns that property's value. However, if *TagName* parameter specifies an object, this method returns the value of that object's **Value** property. Example:

```
Sub Button1_Click()
  ' When clicking this button
  ' retrieves a Tag value
  ' executed on a Data Server
  X = Application.GetValue("DataServer1.InternalTag1")
End Sub
```

4.1.2.16 IsAppRunning

IsAppRunning(*ProcessId*)

Indicates if an application started by the **ExecuteExternalApp** method is currently

executing. Returns True if an application identified on the operating system by *ProcessId* is executing. Otherwise returns False. Example:

```
Sub CommandButton1_Click()
  Application.ExecuteExternalApp _
    "www.elipse.com.br", "", "", 1, processID
  While Application.IsAppRunning(processID)
    ' Waits for this application to end
  Wend
  MsgBox "This application ended!"
End Sub
```

NOTE: The *ProcessId* parameter is the same value appearing on Windows Task Manager, on **PID** column.

4.1.2.17 IsUserMemberOfGroup

IsUserMemberOfGroup(*GroupName*[, *UserName*])

This method checks if a user belongs to a certain group. It has the following parameters:

- **GroupName**: Name of a user group to check
- **UserName**: Name of a user to check. If this parameter is omitted, or if it is an empty **String**, this method considers the user currently logged in Viewer

This method returns True if a user belongs to the *GroupName* group or False otherwise.

4.1.2.18 IsWebViewer

IsWebViewer()

Checks if an application is currently viewed by WebViewer. This method returns True if an application is currently executed on WebViewer. Otherwise, returns False.

4.1.2.19 LoadFormulaDlg

LoadFormulaDlg(*FormulaName*[, *UnitName*[, *ValueName*]])

This method presents a dialog box that allows users to select a value set and a destination Unit, when loading a Formula. This method has the *FormulaName* parameter, which determines the name of a Formula object to process.

Use the **LoadFormulaDlg** method to call a dialog box for loading data from a Formula object specified by *FormulaName*. On this dialog box it is possible to specify which value set (*UnitName*) is sent to which Tag set (*ValueName*). On this message box a user has all value and Unit sets available in a Formula object, and can freely set one to each other. When a user clicks **OK**, the value set is loaded on

the specified Unit. Example:

```
Sub Button1_Click()
  ' Calls the dialog box to operate
    Dim val
    Application.LoadFormulaDlg("Formula1")
End Sub
```

4.1.2.20 LoadFormulaValues

LoadFormulaValues(*FormulaName*, *UnitName*, *ValueName*)

This method automatically loads a value set to a destination Unit, displaying a dialog box that allows users to inform different values from the ones defined on a Formula. This method has the following parameters: *FormulaName* determines the name of a Formula and *UnitName* determines the name of a Unit. The name of the value set is configured in the *ValueName* parameter.

A message box is displayed and allows users to inform values different than the ones defined for each Formula values.

NOTE: This method returns a logical value, that is, returns True if successful and False on failure, which does not mean a script error.

Example:

```
Sub Button1_Click()
  Application.LoadFormulaValues "Formula1",
    "Unit1", "Value1"
End Sub
```

4.1.2.21 LoadFormulaValuesQuiet

LoadFormulaValuesQuiet(*FormulaName*, *UnitName*, *ValueName*)

Loads a value set to a destination Unit, without any message. This method has the following parameters: *FormulaName* determines the name of a Formula and *UnitName* determines a Unit name. The name of the value set is configured in the *ValueName* parameter. Example:

```
Sub Button1_Click()
  Application.LoadFormulaValuesQuiet "Formula1",_
    "Unit3", "Value1"
End Sub
```

NOTE: This method is also accessible through the **Formula** object.

4.1.2.22 LoadReport

LoadReport(*ReportName*)

Loads a Report template. The *ReportName* parameter is the name of a Report to load. Example:

```
Sub Rect_Click()
  ' Loading a pre-defined report
  Set strRep = Application.LoadReport("[Report3]")
  strRep.PrintPreview ' Previewing print
End Sub
```

4.1.2.23 Login

Login([*Mode*])

Opens a dialog box for *login* (user authentication) in an application. The logged-in user remains in memory until another *login* or *logout* (when a user quits an application) is performed. This method has the *Mode* parameter, which is a **Boolean** determining whether a confirmation or failure message must be displayed (default is False). When a Screen is about to open (using the **OpenScreen** method), this method checks if there is any security settings. If True, that Screen is only opened if the logged-in user has permissions. Otherwise, a *login* dialog box is opened.

4.1.2.24 LoginUser

LoginUser(*Username*, *UserPassword*)

Executes a *login* for a specific user without displaying any message. The *Username* parameter is a user name and the *UserPassword* parameter is the password for this user. This method returns True if user *login* was successful and False otherwise. If the user passed in the *Username* parameter is configured to change the password on the next *login*, this method returns False.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.25 Logout

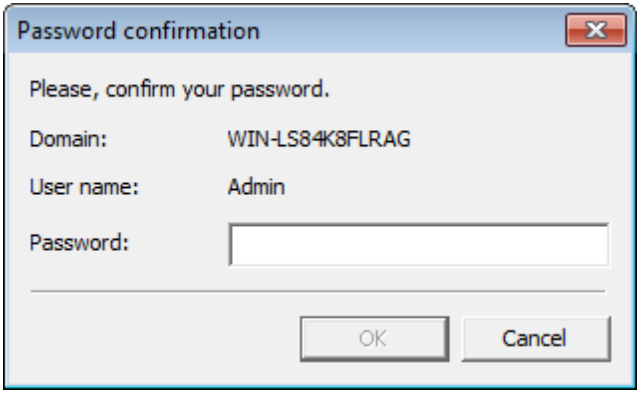
Logout([*Mode*])

Executes a *logout* (when the current user quits an application) from Viewer. In case there is no logged-in user, this method has no effect. From this moment on, it is considered that an "Anonymous" user is using that application. (Users can use the **OnLogout** event to perform a script to go to its initial Screen or end that application). This method has the optional *Mode* parameter, which is a **Boolean** determining whether a confirmation or failure message must be displayed (default is False).

4.1.2.26 PasswordConfirm

PasswordConfirm(*Mode*)

This method opens a dialog box asking the currently logged-in user to retype the password. It returns True if password is confirmed or False otherwise. The **Boolean** *Mode* parameter determines whether a *logout* must be performed in case of failure.



Password confirmation

If this dialog box is closed by clicking **Cancel**, this method returns False. In case there is no logged-in user, this method returns False without opening a dialog box. In case the typed password is wrong, it is then asked three more times. If users type it incorrectly these three times, this dialog box is then closed and this method returns False.

NOTE: This method is only available if the logged-in user belongs to an E3 Domain.

4.1.2.27 Playsound

Playsound(*Filename*)

Plays a sound file whose path and name are indicated in the *Filename* parameter. This file must follow these specifications:

- It must be in a Windows sound format (.wav extension)
- If this file is in a project (added using the **Insert resource** option), its name must be enclosed in brackets
- If a folder was created on a project and this file was added using the **Insert resource** option, this path must be enclosed in quotation marks (for example, "c:\sound\ding.wav"). If this file is on a local directory, its name does not need to be enclosed in quotation marks, it is enough to type a path (for example, c:\sound\ding.wav).

Example:

```
Sub InitialScreen_OnAlarm()
  ' If there is an active alarm, an alert sound is played.
  ' When this alarm is recognized, sound stops.
  Set Alarm = Application._
    GetObject("AlarmConfig1.Area1.AlarmSource1")
    If Alarm.ActiveNACKAlarms = True Then
      Application.PlaySound("[ringin.wav]")
  End If
End Sub
```

4.1.2.28 SelectMenu

SelectMenu(*Menu*[, *Left*[, *Top*]])

This method displays a *pop-up* menu as specified by the *Menu* parameter. This parameter is a text with several options delimited by a vertical bar, each one of these **Strings** is a menu option. In case there is a set of two consecutive delimiters, a separator is inserted. Use opening and closing braces to create a submenu. An asterisk in front of a **String** indicates an already selected option. An exclamation point disables that option.

Dialog box's position can be set through the *Left* and *Top* parameters, which indicate the distance from Screen's left margin and top in pixels, respectively. In case these parameters are not informed, this menu is positioned according to the region where the mouse pointer was clicked.

This method returns 0 (zero) if no option is selected, or an option number, 1 for the first option, 2 for the second, and so on. Example:

```
Sub Button1_Click()
  op = _
    Application.SelectMenu(_
      "Option1||Option2{*Option2|Option3}|Option4|!Opption5")
  If op = 1 Then
    MsgBox "Option 1 was selected"
  ElseIf op = 2 Then
    MsgBox "Option 2 was selected"
  ElseIf op = 3 Then
    MsgBox "Option 3 was selected"
  ElseIf op = 4 Then
    MsgBox "Option 4 was selected"
  ElseIf op = 0 Then
    MsgBox "No option was selected"
  End If
End Sub
```

4.1.2.29 SetValue

SetValue(*TagName*, *NewVal*)

This method sets the value of an object inside server. The **SetValue** method searches for an object or property executed on a server and sets the value specified in the *TagName* parameter. The type and value of the *NewVal* parameter must be supported by the object specified in *TagName*. Example:

```
Sub Button1_Click()
  ' When clicking Button1 sets value 20
  ' to Value property of the tag
  Application.SetValue "DataServer1.InternalTag1", 20
End Sub
```

4.1.2.30 ShowDatePicker

ShowDatePicker(*DateValue*, *Left*, *Top*[, *DefaultDate*])

Opens a dialog box to change date and time. This method returns True if users confirm a date or False if users cancel editing. The new date is returned in the *DateValue* parameter. Dialog box's position can be configured using the *Left* and *Top* parameters, which indicate the distance from Screen's left margin and top in pixels, respectively. In case these parameters are not informed, this dialog box is centralized on the screen. The value of the *DefaultDate* parameter is the initial date and time when this dialog box was opened. If no date is informed, the application assumes the current date. If no time is informed, it starts as "00:00:00". If no date nor time are informed, it starts with the current date and time. Example:

```
Sub Text2_Click()
  Dim newTime
  Application.ShowDatePicker newTime, 300, 300
  MsgBox "The time is : " & newTime
End Sub
```

4.1.2.31 ShowFilePicker

ShowFilePicker(*Open*, *FileName*[, *Extension*[, *Flags*[, *Filter*]]])

Displays Windows **Save** and **Open File** dialog boxes. The *Open* parameter indicates a type of dialog box to open. If True, opens the **Open File** dialog box. If False, opens the **Save** dialog box. The *FileName* parameter indicates a variable to store the file name to save or load, in case this method returns True. This parameter must be a variable. The *Extension* parameter is optional and informs the default file extension to attach to the file name on this dialog box, when no extension is informed. In case it is empty, no extension is attached at the end of the file name. Multiple extensions can be specified by using a semicolon as a delimiter. This **String** must end with double vertical bars (||).

The *Flags* parameter is optional and defines dialog box's behavior. This is an integer corresponding to the sum of values from the next table. The *Filter* parameter is

optional and defines a set of **String** pairs specifying filters that can be applied to files. The first **String** describes a filter and the second one indicates a type of extension to use.

Possible combinations for Flags parameter

VALUE	DESCRIPTION
1	**CREATEPROMPT**: If users specify a non-existing file, this flag allows asking about the creation of this file. If users choose to create this file, this dialog box is closed and the file name is returned in the *Filename* parameter. Otherwise, this dialog box remains open.
2	**FILEMUSTEXIST**: Specifies that users can only type existing file names. Otherwise, this dialog box displays a warning on the message box.
4	**NOCHANGEDIR**: Recovers the current directory to its original value in case users have changed the directory while searching for a file. It has no effect for **Open File** dialog box on Windows XP.
8	**NODEREFERENCELINKS**: Configures this dialog box to return the selected shortcut file (.lnk). If this flag is not specified, this dialog box returns the path and file name referenced by the shortcut file.
16	**NOREADONLYRETURN**: Determines that the returned file is not read-only, and the directory is not record-protected.
32	**PATHMUSTEXIST**: Specifies that users can only inform valid files and directories, otherwise a message box is displayed to warn users.
64	**READONLY**: Allows that the **Read Only** option to be initially selected when this dialog box is created.
128	**OVERWRITEPROMPT**: Allows that the **Save As** dialog box generates a message informing that this file exists, and asks for confirmation to overwrite that file.

Filter example:

```
"Chart Files (*.xlc)|*.xlc|Excel spreadsheets (*.xls)_
  |*.xls|Data Files (*.xlc;*.xls)|*.xlc; *.xls
  |All files (*.*)|*.*||"
```

4.1.2.32 ShowPickColor

ShowPickColor(*ColorValue*[, *Color*, *Left*, *Top*])

Opens Windows **Colors** dialog box to select a color. The decimal value of the selected color is returned in the *ColorValue* parameter. The *Color* parameter indicates a previously selected color on the color palette. If this parameter is not informed, assumes the value 0 (zero, black). This dialog box's position can be configured using the *Left* and *Top* parameters, which indicate, respectively, the distance from Screen's left margin and top, in pixels. In case these parameters are not informed, this dialog box is centered. Example:

```
Sub CommandButton_Click()
  Dim newColor
  Dim defaultColor
  defaultColor = 65280 ' Light green
  If Application.ShowPickColor(newColor,_
    defaultColor, 90, 90) Then
    Screen.Item("Rectangle1").ForegroundColor = newColor
    Screen.Item("Text1").Value = newColor
  End If
End Sub
```

4.1.2.33 Stopsound

Stopsound()

Stops a sound that is executing.

4.1.2.34 ToggleValue

ToggleValue(*TagName*, *ValA*, *ValB*)

The **ToggleValue** method searches for an object value or property in execution on a server and compares it to *ValA* and *ValB* parameters. If the searched value is equal to *ValB*, the object or property specified in *TagName* receives the value of *ValA*. Otherwise, it receives the value of *ValB*. In case the value of *TagName* is not *ValA* nor *ValB*, the **ToggleValue** method sets the value specified in *ValA*. Example:

```
Sub Button1_Click()
  ' When clicking the button, sets the value
  ' to a Tag executed on a Data Server.
  ' Sets a value of 20 to the Tag.
  Application.SetValue "DataServer1.InternalTag1", 20
  ' As the value of InternalTag1 already is 20,
  ' the ToggleValue method alternates the value to 30.
  Application.ToggleValue "DataServer1.InternalTag1", 30, 20
End Sub
```

4.1.2.35 TrackEvent

TrackEvent(*EventMessage*, *Comment*, *TimeStamp*)

The **TrackEvent** method allows generating events manually via script. These events can be generated on Viewer as well as on Server, and are registered on an application's database table.

Parameters of the TrackEvent method

NAME	DESCRIPTION
EventMessage	Contains an event's message (maximum of 200 characters).
Comment	(Optional) Contains additional comments about this event (maximum of 200 characters).
TimeStamp	(Optional) Indicates the date and time this event occurred. If it is not specified, E3 assumes the current date and time.

The **TrackEvent** method only records events in case the **Events Recording** option in Domain Options is enabled. Events are recorded on a database table, which is also defined on the Event Recording settings.

For more information on Domain's Event Log, please check the **E3 User's Manual**. Example:

```
Sub Button1_Click()
  Dim Tag, User, Comment
  Set Tag = Application.GetObject("IO.Inputs.IO01")
  If Application.ESign(Tag.PathName, , "Value Change", _
    Tag.Value, 1, User, Comment) Then
    If Tag.WriteEx 1 Then
      Application.TrackEvent _
        "Tag IO.Inputs.IO01 changed to 1 " &_
        "by the user" & User, Comment
    End If
  End If
End Sub
```

4.1.2.36 UserAdministration

UserAdministration()

This method opens a dialog box that allows editing a list of Server's users. The available functions are:

- Display a list of all users
- Delete users (it is not possible to delete the current user)

- Add and edit users
- Edit user settings
- Change user's password
- Change other user's data (login, name, etc.)

IMPORTANT: Only an Administrator has access to the **UserAdministration** method. A dialog box for user configuration is only accessible to a user enabled as Administrator.

4.1.3 Properties

This section contains information about the properties of the Viewer object.

4.1.3.1 BlinkTime

9 Defines a time, in milliseconds, between each state change when an object must blink, that is, each time an object has a Link and the **Blink** option is selected. Default value of this property is 200 ms.

NOTE: The minimum time for Screen refreshing is 55 ms. Therefore, if this property is set with a time less than that, this configuration has no effect at all.

4.1.3.2 CacheEnable

◩ Keeps in memory all Screens already opened and instantiated on Viewer, allowing a quick switching among them. If this property is enabled, then Screen's cache is also enabled.

4.1.3.3 Caption

A Determines the name of the application that is using Viewer. Default value of this property is an empty **String**.

4.1.3.4 CenterWindow

◩ When enabled, determines that Viewer's window must start centered. Otherwise, default settings are used. Default value of this property is True.

4.1.3.5 CloseButton

◩ If this option is set to True, a **Close** button is enabled on Viewer, and it can be used. Otherwise, this button does not display on its window. Default value of this property is True.

4.1.3.6 CommErrorBkColor

Property used to define a background color for a SetPoint when a Link or Connection fails. Default value of this property is red (**RGB(255, 0, 0)**). Also check the **EnableCommError** property.

4.1.3.7 CommErrorText

A Property used to define an alert message when a Link or Connection fails. Default value of this property is "???". Also check the **EnableCommError** property.

4.1.3.8 CommErrorTextColor

Property used to define a text color for a SetPoint when a Link or Connection fails. Default value of this property is yellow (**RGB(255, 255, 0)**). Also check the **EnableCommError** property.

4.1.3.9 DisableTaskSwitching

If it is set to True, disables window switching on Viewer. Otherwise, window switching is enabled. Default value of this property is False. This property can be changed at run time by using the **SetDisableTaskSwitching** method.

4.1.3.10 EnableCommError

Enables or disables viewing communication errors. For example, if there is a SetPoint on a Screen that is linked to an I/O Tag, and communication between E3 and this Tag fails, the text configured in the **CommErrorText** property is displayed in that SetPoint, with a color configured in the **CommErrorTextColor** property and a SetPoint's background color defined in the **CommErrorBkColor** property. Default value of this property is True.

4.1.3.11 EnableHeartbeat

Enables or disables sending a *heartbeat* message (sent at fixed intervals, which indicates that the server is active) between a Viewer and a server. In case Viewer stops receiving *heartbeat* messages, this means that a problem occurred, therefore connection must be aborted. Default value of this property is False.

4.1.3.12 EnableInactivity

Enables or disables checking user's inactivity period. For more information, please check Viewer's **OnInactive** event. Default value of this property is False. This property cannot be changed at run time.

4.1.3.13 EnableZoomMenu

◩ If it is set to True, enables displaying a configuration menu for Screen's zoom using the right mouse button at run time, except when a script is configured with a different information on **MouseDown** or **MouseUp** events. Otherwise, this menu is not displayed. Default value of this property is True.

4.1.3.14 HeartbeatPeriodMs

9 Indicates an interval, in milliseconds, between *heartbeat* messages sent by a Server. A *heartbeat* message is always sent when a server remains a period indicated by this property without sending messages for a Viewer. Default value of this property is 2000 (two seconds).

4.1.3.15 HeartbeatTimeoutMs

9 Indicates a time, in milliseconds, that a Viewer waits without receiving messages from a Server. If this time passes and no message is received, Viewer assumes that connection was lost and starts the reconnection process. This time must be greater than the time set in the **HeartbeatPeriodMs** property, preferably greater than the double. Default value of this property is 5000 (five seconds).

4.1.3.16 InactivityTime

9 Defines a maximum waiting time for a mouse or keyboard event before an inactivity period, in minutes. For more information, please check Viewer's **OnInactive** event. Default value of this property is 5 (five) minutes. Example:

```
Sub CommandButton3_Click()
  MsgBox "This application will remain inactive for " & _
    Application.InactivityTime & " minute(s)."
End Sub
```

4.1.3.17 InitialScreen

A Indicates an initial Screen or Frame displayed when Viewer is opened. By using the **WindowStyle** property, users can determine if this window must start maximized, windowed, or minimized. Default value of this property is "InitialScreen".

4.1.3.18 IsPlaybackMode

◩ If True, indicates that Viewer is in execution inside an E3Playback, in *playback* mode. This is a read-only property and it is only available at run time.

4.1.3.19 IsReadOnly

◩ If set to True, indicates that Viewer is in **Read-Only** mode (restricted access).

4.1.3.20 LoginRetries

9 Specifies the number of Viewer's *login* retries, that is, how many times the *login* dialog box is displayed besides the first time. Default value of this property is 2 (two).

4.1.3.21 MaximizeButton

◩ If this property is set to True, a **Maximize** button is enabled on Viewer and this button can be used. Otherwise, this button does not display on that window. Default value of this property is True.

4.1.3.22 MinimizeButton

◩ If this property is set to True, a **Minimize** button is enabled on Viewer and this button can be used. Otherwise, this button does not display on that window. Default value of this property is True.

4.1.3.23 Params

A This property is a vector of key-value pairs, which returns parameters passed to Viewer through the **-params** command prompt option. All values are returned as **Strings**. For example, if Viewer's command prompt contains the following parameters:

```
Viewer -params Language=ENU
```

Users can use the following code to check what is Viewer's starting language.

```
Sub InitialScreen_OnStartRunning()
  Select Case Application.Params("Language")
    Case "ENU"
      Item("Text1").Value = "English"
    Case Else
      Item("Text1").Value = "Unrecognized language"
  End Select
End Sub
```

NOTE: A key's **String** is not case-sensitive (it may be "Language" or "language"), but returned values, specially if used with a **Select** command, are case-sensitive.

4.1.3.24 ReconnectDialogDelaySec

9 Indicates the number of seconds that Viewer waits during a possible reconnection to a server, before displaying a message warning users about this action (this property does not affect the first connection). If it is equal to 0 (zero), a reconnection message is always displayed. To avoid displaying this message, it is advisable to set a very large number (one billion, for example).

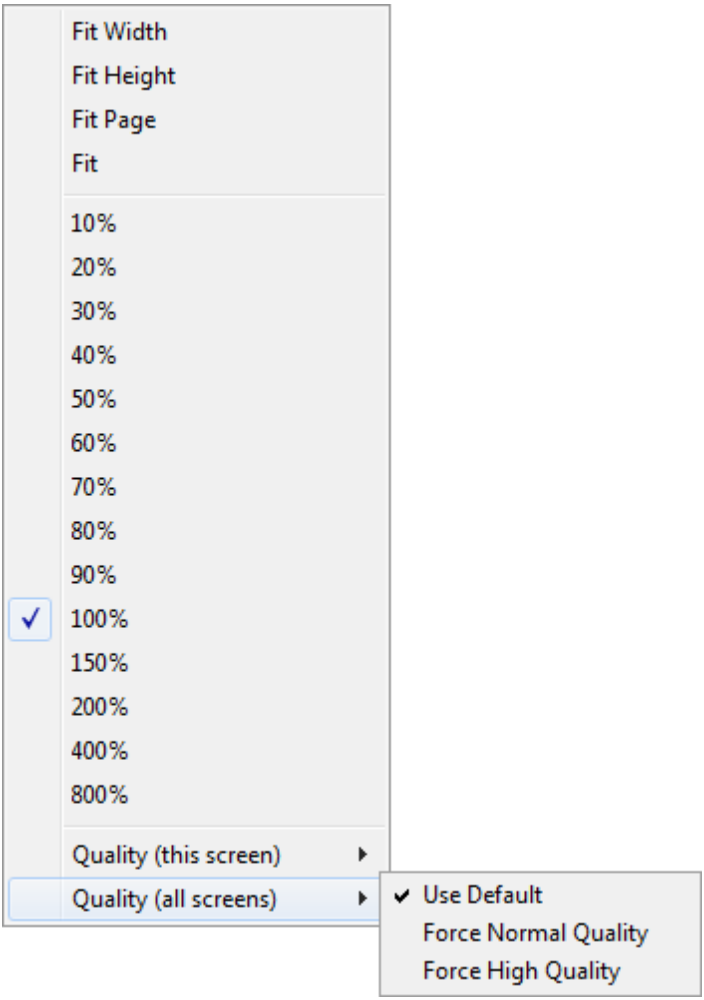
NOTE: When reconnection occurs silently, all windows from the active Viewer are disabled and an hourglass is displayed, indicating that this application is unavailable. During reconnection time, users are not allowed to cancel this process.

4.1.3.25 RenderQuality

Controls drawing quality of all Screens, only if the value of **RenderQuality** property of all Screens is equal to **0 - rqDefault**. Possible values for this property are the following:

- **0 - rqDefault**: Uses a normal quality mode, known as GDI (*Graphics Device Interface*). Corresponds to the **Use Default** item on Viewer's contextual menu **Quality (all screens)**, at run time. This is a default value for applications created on versions prior to 4.0
- **1 - rqNormal**: Forces a normal quality mode (GDI) for drawing all Screens. Corresponds to the **Force Normal Quality** item on Viewer's contextual menu **Quality (all screens)**, at run time
- **2 - rqHighQuality**: Forces a high quality mode (GDI+) for drawing all Screens. Corresponds to the **Force High Quality** item on Viewer's contextual menu **Quality (all screens)**, at run time. This is a default value for applications starting at version 4.0

The next figure displays Viewer's contextual menu at run time, with the respective configuration options for this property.



Viewer's contextual menu at run time

4.1.3.26 ShowKeyPadOnEdit

◩ This property automatically enables displaying Elipse KeyPad, whenever a Screen object that allows edition receives the focus.

4.1.3.27 TargetDPIX

9 Defines a dots-per-inch value, horizontally, of the destination computer's monitor. Default value of this property is -1 (minus one), which means that the current computer's value is assumed.

4.1.3.28 TargetDPIY

9 Defines a dots-per-inch value, vertically, of the destination computer's monitor. Default value of this property is -1 (minus one), which means that the current computer's value is assumed.

4.1.3.29 TargetMarginX

9 Defines the number of pixels that must be discounted from Screen's horizontal resolution (Viewer's working area). Default value of this property is -1 (minus one), which assumes Viewer's window configuration (with or without a title bar, with or without a border) together with the current computer's configuration (Windows-defined border width and title bar).

4.1.3.30 TargetMarginY

9 Defines the number of pixels that must be discounted from Screen's vertical resolution (Viewer's working area). Default value of this property is -1 (minus one), which assumes Viewer's window configuration (with or without a title bar, with or without a border) together with the current computer's configuration (Windows-defined border width and title bar).

4.1.3.31 TargetResolutionX

9 Defines Screen's horizontal resolution for which this application is designed to, in pixels. Default value of this property is -1 (minus one), which assumes the current computer's resolution.

4.1.3.32 TargetResolutionY

9 Defines Screen's vertical resolution for which this application is designed to, in pixels. Default value of this property is -1 (minus one), which assumes the current computer's resolution.

4.1.3.33 TitleBar

◩ If this property is enabled, Viewer's title bar is displayed, according to the **Caption** property. Otherwise, it is hidden. Default value of this property is True.

4.1.3.34 User

A Contains the name of the user currently using Viewer. This is a read-only property.

4.1.3.35 ViewerLanguageId

9 Returns the current Viewer's language code. Possible values for this property are described on the next table. This is a read-only property and it is only available at run time.

Available values for ViewerLanguageId

DECIMAL	HEXADECIMAL	LANGUAGE
4	0x0004	Chinese Simplified
1031	0x0407	German
1033	0x0409	English (US)
1034	0x040A	Spanish
1046	0x0416	Brazilian Portuguese

NOTE: This property **DOES NOT** correspond to Windows installation language and neither the language configured on Windows **Regional and Language Options** (**Clock, Language and Region** on Windows 7) Control Panel.

4.1.3.36 WindowBorder

◩ Enables or disables a border around Viewer's window. Default value of this property is True. This is a read-write property, but changing its value at run time does not change settings for an already opened Viewer, only for windows opened using Viewer's settings.

4.1.3.37 WindowHeight

A Determines the height of Viewer's window, in pixels. Default value of this property is 300.

4.1.3.38 WindowMovable

◩ Indicates whether this window can be moved. Default value of this property is True. This is a read-write property, but changing its value at run time does not change settings for an already opened Viewer, only for windows opened using Viewer's settings.

4.1.3.39 WindowResizable

◩ Indicates whether a window can be resized. This property is effective only if the **WindowBorder** property is set to True. Default value of this property is True. This is a read-write property, but changing its value at run time does not change settings for an already opened Viewer, only for windows opened using Viewer's settings.

4.1.3.40 WindowSmallTitle

Indicates whether Viewer's window must have a small title bar. This property is effective only if the **TitleBar** property is True. Default value of this property is False. This is a read-write property, but changing its value at run time does not change settings for an already opened Viewer, only for windows opened using Viewer's settings.

4.1.3.41 WindowStayOnTop

Indicates whether Viewer's window must always be on top of other windows. Default value of this property is False. This is a read-write property, but changing its value at run time does not change settings for an already opened Viewer, only for windows opened using Viewer's settings.

4.1.3.42 WindowStyle

Defines an initial style for Viewer's window. Available options are:

- **0 - Maximized**: Viewer starts maximized
- **1 - Windowed**: Viewer starts windowed
- **2 - Minimized**: Viewer starts minimized

4.1.3.43 WindowWidth

Determines the width of Viewer's window, in pixels. Default value of this property is 400.

4.2 Frames and Splitters

This section contains information about **methods** and **properties** of the Splitter object and about **properties** of the Frame object. The Splitter object does not have events associated to it and the Frame object does not have events nor methods associated to it.

4.2.1 Splitter Methods

This section contains information about the methods of the Splitter object.

4.2.1.1 BringToFront

BringToFront()

Brings to the front a Splitter that is hidden or under another one.

4.2.1.2 CaptureScreen

CaptureScreen(*Filename*)

Captures the content of a Splitter, saving it to a file pointed by *Filename*, in BMP format. Example:

```
Sub CommandButton1_Click()
  ' When clicking this button,
  ' copies Splitter's content
  ' to Frame.bmp file.
  Screen.Frame.CaptureScreen ("c:\temp\frame.bmp")
End Sub
```

4.2.1.3 Close

Close(*Code*)

Use the **Close** method to close a Frame window. The *Code* parameter contains the returned value for the **DoModal** method, if this window was called by this method. Example:

```
Sub CloseButton_Click()
  ' When clicking this button,
  ' closes the window.
  Screen.Close(0)
End Sub
```

4.2.1.4 FlashWindow

FlashWindow(*Number*, *Time*)

This method forces Viewer's icon on Windows Notification Area to start flashing. The *Number* parameter determines the number of times Viewer's icon must flash and *Time* determines a time (in milliseconds) between flashes. Example:

```
Sub Text1_Click()
  Set frame = Application.GetFrame("_top")
  frame.FlashWindow 50, 500
End Sub
```

4.2.1.5 MaximizeFrame

MaximizeFrame()

Maximizes a Frame or modal Screen.

4.2.1.6 MinimizeFrame

MinimizeFrame()

Minimizes a Frame or modal Screen.

4.2.1.7 MoveFrame

MoveFrame([*PosX*,][*PosY*,][*SizeX*,][*SizeY*])

Moves and resizes a Frame to a specific coordinate and size. The *PosX* and *PosY* parameters inform a new position, in pixels, relative to the left and top, respectively. The *SizeX* and *SizeY* parameters inform a new size and height,respectively, in pixels or Himetric. All parameters are optional. Example:

```
Sub Screen2_OnPreShow(vArg)
  ' When Screen2 is opened on Test frame,
  ' changes Frame's
  ' position and size
  Application.GetFrame("Test").MoveFrame 100, 100, 200, 200
End Sub
```

NOTE: Size values for this method can be informed as numbers or **Strings**. In case of numbers, they are considered as pixels. In case of **Strings**, if they are followed by an "hm" unit, they are interpreted as Himetric. Any other case is considered as pixels.

4.2.1.8 OpenScreen

OpenScreen(*ScreenName*, *Arg*)

The **OpenScreen** method opens a Screen inside a Splitter. The *ScreenName* parameter determines the name of a Screen to open. Users can also specify a zoom percentage for a Screen and enable a scroll bar by using the "?" key, such as in the following template.

```
<screen-name>?<zoom>?<enable-scroll>
```

Where *screen-name* is the name of a Screen to open, *zoom* is a zoom percentage, and *enable-scroll* enables or disables scrolling. The zoom percentage of a Screen can have the following values:

- **1**: The entire page
- **2**: This Screen width occupies 100% of Splitter's width, with proportional height
- **3**: This Screen height occupies 100% of Splitter's height, with proportional width
- **4**: This Screen completely fills a Splitter
- **5 to 100**: It is the equivalent of a zoom percentage of the Screen itself

Enabling a scroll can have the following values:

- **0**: Disables scrolling

- **1**: Enables scrolling

The *Arg* parameter allows passing the specified value to a Screen, by using the **OnPreShow** event. Example:

```
Sub Button1_Click()
  ' When clicking the button opens Screen2 on Test frame
  ' and passes 1 to use on OnPreShow event
  Application.GetFrame("Test")._
    OpenScreen "Screen2?100?0", "This is a test."
End Sub

Sub Screen2_OnPreShow(vArg)
  ' This message box displays the sentence
  ' "This is a test."
  MsgBox vArg
End Sub
```

4.2.1.9 Refresh

Refresh(*Force*)

The **Refresh** method allows forcing a redrawn of a Screen's or Splitter's content. It must be used in Viewer scripts with massive processing (loops, for example) or in method calls demanding a long time and also demanding a visual indication for the progress of a process to users.

Due to the general redrawn being a heavy operation, default version of the **Refresh** method (without parameters) is optimized to ignore redrawn requests from E3. This standard behavior is ideal for loop progress indications, where a lot of redrawing is performed. The *Force* parameter disables this optimization, ensuring that for each call to **Refresh** method, a redrawn is performed. However, when using this option, the **Refresh** method cannot be called repeatedly, such as inside a loop. Example:

```
Sub CommandButton1_Click()
  ' Draws a progress bar for an operation
  While i < 31
    Screen.Item("Rectangle2")_
      .HorizontalPercentFill = (i / 30) * 100
    Frame.Refresh True
    ' <-- some lengthy operation -->
  Wend
End Sub
```

4.2.1.10 RestoreFrame

RestoreFrame()

Allows restoring a Frame window to its original size.

4.2.1.11 SetDisableTaskSwitching

SetDisableTaskSwitching(*Disable*)

Enables or disables window switching. The *Disable* parameter is a **Boolean** value that indicates whether a window switching is enabled or not. This method updates Viewer's **DisableTaskSwitching** property.

NOTES:

- If more than one Viewer instance is executing on the same machine, and at least one of these instances use the **SetDisableTaskSwitching** method, that change affects all Viewers on this machine
- If users want to change window's title or style, the **SetFrameOptions** method must be used after calling the **SetDisableTaskSwitching** method

4.2.1.12 SetForegroundWnd

SetForegroundWnd()

The **SetForegroundWnd** method activates and moves the focus to a Viewer window. This method is useful when users want to grab operator's attention for an event, when Viewer window is hidden or minimized.

4.2.1.13 SetFrameOptions

SetFrameOptions(*Title*, *Flags*)

Used to configure Frame's title on a window and that window's style. The *Title* parameter is a **String** that contains window's title. This text is displayed if Screen's **Caption** property is empty.

The *Flags* parameters specifies window's style. If this parameter is omitted, its default value is -1 (minus one). This value is used to keep window's previous configuration. When this value is not -1 (minus one), users can change window's style by specifying a sum of values of all combinations described on the next table.

Possible combinations for the Flags parameter

VALUE	DESCRIPTION
1	Enables a title bar on this window
2	Enables a **Close** button on this window
4	Enables a **Minimize** button on this window
8	Enables a **Maximize** button on this window
16	Enables a border on this window
32	Specifies that this window can be resized (to do so, this window must have a border)

VALUE	DESCRIPTION
64	Specifies that this window can be moved
256	Specifies that this window stays on top of a Screen
512	Specifies that this window is configured in a **Tool Bar** style
1024	Disables object buttons
2048	Centers this window

Example:

```
Sub Screen_OnPreShow()
  Frame.SetFrameOptions("Alarm Screen", 114)
End Sub
```

In the previous example, the value 114 (2 + 16 + 32 + 64) indicates that this window has a **Close** button enabled (2), a border (16), can be resized (32), and can be moved (64). This window's title is "Alarm Screen".

In the **Open Screen** and **Open Modal Screen** Picks, users can also configure a window style during edition, by using the Window Style dialog box. For more information, please check topic **Picks**.

NOTE: The **SetFrameOptions** method must be used after calling the **SetDisableTaskSwitching** method, if users want to change window's title or style.

4.2.1.14 ShowScreen

ShowScreen(*Screen*, *Zoom*, *Scrollbars*[, *Arg*])

This method loads a Screen on a Splitter, then closing the Screen or Frame previously loaded. The available parameters on this method are the following:

- **Screen**: A Screen object to load. This object must be returned by Viewer's **GetScreen** method
- **Zoom**: Screen's zoom percentage
- **Scrollbars**: A **Boolean** that indicates whether this Screen's scroll bar must be enabled or not
- **Arg**: An optional **String** whose value is used on the **OnPreShow** event of the Screen to load

The **SplitLink** property is updated after calling this method.

NOTE: The *Screen* parameter only accepts an object returned by Viewer's **GetScreen** method. This method fails if using a Screen object returned by another method or property, such as the **Screen** property, common to all Screen objects.

4.2.2 Splitter Properties

This section contains information about the properties of the Splitter object.

4.2.2.1 IsHTML

The **IsHTML** property returns True if a Splitter contains HTML code inserted into a Frame. Otherwise, returns False.

4.2.2.2 SplitBorder

Enables or disables a Splitter border and determines whether a border between main and remaining Splitters must be displayed at run time. This property has no effect on the remaining Splitter. Default value of this property is True.

4.2.2.3 SplitDockPosition

Indicates Splitter's position on a Screen. Available options are described on the next table.

Available options for SplitDockPosition

OPTION	DESCRIPTION
0 - dockRemaining	Places this Splitter as the remaining one, that is, it occupies the remaining space on horizontal or vertical Splitter.
1 - dockTop	Places this Splitter as the main one, above the remaining.
2 - dockBottom	Places this Splitter as the main one, below the remaining.
3 - dockLeft	Places this Splitter as the main one, on the left side of the remaining.
4 - dockRight	Places this Splitter as the main one, on the right side of the remaining.

4.2.2.4 SplitLink

The **SplitLink** property contains a link that can be displayed on a Splitter. Users can specify a project Screen, an executable, or an Internet link. In case of Screens, users can specify a zoom percentage and enable scroll bars using a "?" key, such as in the template "<screen-name>?<zoom>?<scroll-bar>", where *screen-name* is the name of a Screen to open, *zoom* is a zoom percentage, and *scroll-bar* is 1 (one) to enable or 0 (zero) to disable. The *zoom* and *scroll-bar* parameters are valid only if the indicated link is a Screen. Otherwise, they are ignored. If the *zoom* parameter is

not informed, it assumes 100%. If the *scroll-bar* parameter is not informed, it assumes 1 (one), that is, enabled.

4.2.2.5 SplitResizable

◩ Determines whether the main Splitter can be resized at run time. This property has no effect on the remaining Splitter. Default value of this property is True.

4.2.2.6 SplitValue

9 The **SplitValue** property determines a value to set to a Frame's Splitter. In Studio, this value can be followed by the unit "%" (percentage), "hm" (Himetric), or "px" (pixels). If the unit is omitted, this value is considered in Himetric. At run time, this value must be a number and it is always considered as a percentage.

4.2.3 Frame Properties

This section contains information about the properties of the Frame object.

4.2.3.1 Caption

A The **Caption** property defines a Frame's title to display on Viewer's title bar.

4.3 Screens and Screen Objects

This section contains information about events, methods, and properties of Screens and Screen Objects.

4.3.1 Screen

This section contains information about **events**, **methods**, and **properties** of the Screen object.

4.3.1.1 Events

This section contains information about the events of the Screen object.

4.3.1.1.1 Click

Click()

This event occurs when the left mouse button is pressed on a Screen. This event does not occur if that Screen is not visible or if its **Enabled** property is set to False. Screen visibility depends on three factors: **Visible** property set to True, parent object visible, and object's **Layer** property on Screen's layer. Example:

```
Sub Screen_Click()
  ' Displays a message box when
```

```
  ' a user clicks a screen
  MsgBox "You clicked the screen."
End Sub
```

4.3.1.1.2 DbClick

DbClick()

This event occurs when there is a left mouse button double-click on a Screen. This event does not occur if that object is not visible or if its **Enabled** property is set to False. Object's visibility depends on three factors: **Visible** property set to True, parent object visible, and object's **Layer** property on Screen's layer. Example:

```
Sub Screen_DbClick()
  ' Displays a message box when
  ' a user double-clicks a screen
  MsgBox "You double-clicked the screen."
End Sub
```

4.3.1.1.3 KeyDown

KeyDown(*KeyCode*, *Shift*)

This event occurs when a key is pressed, regardless of Screen focus.

Variables of the KeyDown event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Screen1_KeyDown(KeyCode, Shift)
  ' Displays a message box when
  ' a user presses a key
  MsgBox "Key code: " & KeyCode
End Sub
```

NOTE: For a list with all key codes available for the *KeyCode* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.3.1.1.4 KeyUp

KeyUp(*KeyCode*, *Shift*)

This event occurs when a key is released, regardless of the Screen focus.

Variables of the KeyUp event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

Example:

```
Sub Screen1_KeyUp(KeyCode, Shift)
  ' Displays a message box when a user
  ' releases a key
  MsgBox "Key code: " & KeyCode
End Sub
```

NOTE: For a list with all key codes available for the *KeyCode* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.3.1.1.5 MouseDown

MouseDown(*Button*, *ShiftState*, *MouseX*, *MouseY*)

This event occurs when any mouse button is pressed on a Screen. Use the **MouseDown** event to determine specific actions when a user clicks a Screen.

Variables of the MouseDown event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays Screen's X coordinate where the mouse pointer was clicked.

NAME	DESCRIPTION
MouseY	Displays Screen's Y coordinate where the mouse pointer was clicked.

Example:

```
Sub InitialScreen_MouseDown(Button, ShiftState, MouseX, MouseY)
  ' Quits the application when a mouse click occurs
  ' on the InitialScreen object.
  Application.Exit()
End Sub
```

4.3.1.1.6 MouseUp

MouseUp(*Button*, *ShiftState*, *MouseX*, *MouseY*)

This event occurs when a previously clicked mouse button is released. Use the **MouseUp** event to determine specific actions to be triggered only when a mouse button is released.

Variables of the MouseUp event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays Screen's X coordinate where the mouse pointer was clicked.
MouseY	Displays the Screen's Y coordinate where the mouse pointer was clicked.

Example:

```
Sub InitialScreen_MouseUp(Button, ShiftState, MouseX, MouseY)
  ' Quits the application only when a user releases the mouse
button.
  Application.Exit()
End Sub
```

4.3.1.1.7 OnHide

OnHide()

This event occurs when a Screen is about to be closed. Use the **OnHide** event to

perform any operations before closing that Screen object. This event may occur on several situations:

- When a Screen is replaced by another one, by using the **OpenScreen** or **ShowScreen** methods
- When a user closes a window where that Screen is
- When the **Close** method is used from a Screen object
- When Viewer is closed or quit

Example:

```
Sub InitialScreen_OnHide()
  Application.Exit()
End Sub
```

4.3.1.1.8 OnPreShow

OnPreShow(Arg)

This event occurs before displaying a Screen. The *Arg* variable receives the content of the *Arg* parameter of the **OpenScreen** or **ShowScreen** methods, which generate this event. After that, the **OnShow** event is generated. Example:

```
Sub Screen1_OnPreShow(Arg)
  ' Screen1's title to display
  ' was passed as a parameter when using the OpenScreen
  ' method that generated this event.
  Caption = Arg
End Sub
```

4.3.1.1.9 OnShow

OnShow()

This event occurs in the exact instant that a Screen is displayed. Use the **OnPreShow** event to perform any operation before displaying that Screen. Example:

```
Sub MainScreen_OnShow()
  MsgBox "Welcome to the system!"
End Sub
```

4.3.1.2 Methods

This section contains information about the methods of the Screen object.

4.3.1.2.1 Close

Close(*Code*)

Use the **Close** method to close a Screen. This method generates the **OnHide** event

before being performed. The *Code* parameter contains the value returned by the **DoModal** method, if that Screen is called by this method. Example:

```
Sub CloseButton_Click()
  ' When clicking CloseButton, closes this window
  Screen.Close(0)
End Sub
```

4.3.1.2.2 FromPixelX

FromPixelX(*XPixel*)

Converts Screen's X coordinate, indicated by the *XPixel* parameter, from pixels to Himetric. This method is complementary to the **ToPixelX** method.

4.3.1.2.3 FromPixelY

FromPixelY(*YPixel*)

Converts Screen's Y coordinate, indicated by the *YPixel* parameter, from pixels to Himetric. This method is complementary to the **ToPixelY** method.

4.3.1.2.4 ToPixelX

ToPixelX(*XHimetric*)

Converts Screen's X coordinate, indicated by the *XHimetric* parameter, from Himetric to pixels. This method is complementary to the **FromPixelX** method.

4.3.1.2.5 ToPixelY

ToPixelY(*YHimetric*)

Converts Screen's Y coordinate, indicated by the *YHimetric* parameter, from Himetric to pixels. This method is complementary to the **FromPixelY** method.

4.3.1.3 Properties

This section contains information about the properties of the Screen object.

NOTE: E3 uses the HIMETRIC system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

4.3.1.3.1 BackgroundColor

Specifies Screen's background filling color. In scripts, use VBScript's **RGB** method to gather a color to associate to this property. Default value of this property is gray

(**RGB(192, 192, 192)**).

4.3.1.3.2 Caption

The **Caption** property defines a Screen title to display on Viewer's title bar.

4.3.1.3.3 FillStyle

This property specifies a fill style for Screens and Screen objects. The next table contains valid values for this property.

Available options for FillStyle

OPTION	DESCRIPTION
0 - Solid	Fill is solid (default)
1 - Hollow	No fill. This value is not available for a Screen object
2 - Horizontal	Fill has horizontal stripes
3 - Vertical	Fill has vertical stripes
4 - Downward	Fill has down stripes from left to right in a 45-degrees angle
5 - Upward	Fill has up stripes from left to right in a 45-degrees angle
6 - Cross	Fill has horizontal and vertical stripes
7 - DiagonalCross	Fill has diagonal stripes from left to right in a 45-degrees angle
8 - Gradient	Fill has a gradient using **ForegroundColor** as well as **BackgroundColor**. This effect is defined by the **GradientStyle** property
9 - SemiTransparent	Leaves an object translucent. This value is not available for a Screen object
10 - MouseArea	Fill is empty, but an object still responds to events (default). This value is not available for a Screen object
11 - Background	Fills a Screen with a background color
12 - Picture	Fills a Screen with the picture selected on **PictureFile** property. This value is only available for a Screen object

NOTE: The **FillStyle** property is not available for **Picture** objects.

4.3.1.3.4 ForegroundColor

Specifies Screen's foreground color. In scripts, use VBScript's **RGB** method to gather a color to associate to this property. Default value of this property is black (**RGB(0, 0, 0)**). Applications previous to the introduction of this property have both **ForegroundColor** and **BackgroundColor** properties set to the color stored on the

BackgroundColor property, and its fill style set to **11 - Background**, which paints the whole Screen with the background color (old behavior, before creating styles). Example:

```
Sub Screen1_Click()
  ' Changes foreground color to blue
  ForegroundColor = RGB(0, 0, 255)
End Sub
```

4.3.1.3.5 GradientStyle

This property specifies Screen's gradient filling style. This property is only used when the **FillStyle** property is set to 8 (**Gradient**). Gradients consider a color change as starting at **ForegroundColor** and ending at **BackgroundColor**.

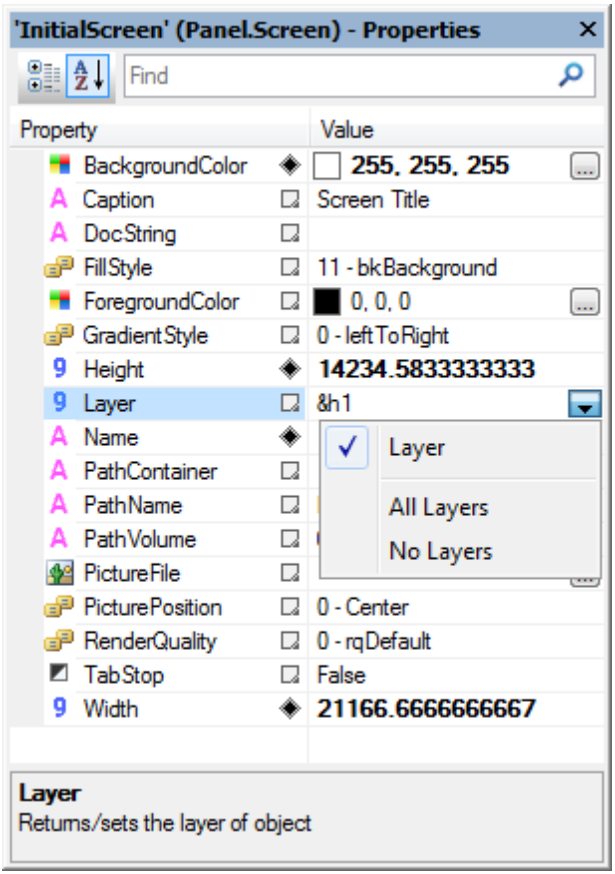
Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient, from left to right
1 - RightToLeft	Vertical gradient, from right to left
2 - VerFromCenter	Vertical gradient, from center to border
3 - VerToCenter	Vertical gradient, from border to center
4 - BottomUp	Horizontal gradient, from bottom to top
5 - TopDown	Horizontal gradient, from top to bottom
6 - HorzFromCenter	Horizontal gradient, from center to border
7 - HorzToCenter	Horizontal gradient, from border to center
8 - DiagUpRight	Diagonal gradient to the top, with foreground color on the right (default)
9 - DiagUpLeft	Diagonal gradient to the top, with foreground color on the left
10 - DiagUpFromCenter	Diagonal gradient to the top, from center to border
11 - DiagUpToCenter	Diagonal gradient to the top, from border to center
12 - DiagDownLeft	Diagonal gradient to the bottom, with foreground color on the left
13 - DiagDownRight	Diagonal gradient to the bottom, with foreground color on the right
14 - DiagDownFromCenter	Diagonal gradient to the bottom, from center to border
15 - DiagDownToCenter	Diagonal gradient to the bottom, from border to center
16 - SpotSouthEast	Gradient with foreground color, from right lower corner
17 - SpotSouthWest	Gradient with foreground color, from left lower corner
18 - SpotNorthWest	Gradient with foreground color, from left upper corner
19 - SpotNorthEast	Gradient with foreground color, from right upper corner

OPTION	DESCRIPTION
20 - SpotFromCenter	Gradient with background color, from center to border
21 - SpotToCenter	Gradient with background color, from border to center

4.3.1.3.6 Layer

9 This property defines in which layers an object must appear. Its value represents a 32-bit mask, one bit for each layer. Therefore, up to 32 layers can be defined. Thus, objects may be logically grouped and displayed or hidden only by modifying the **Layer** property's mask.



Layer options

Example:

```
Sub Screen1_Click()
  Layer = 1
End Sub
```

NOTE: Object's visibility depends on three factors: **Visible** property must be set to True, parent object must be visible, and object's **Layer** property must be enabled for a Screen.

4.3.1.3.7 PictureFile

Contains picture's filename that is used as Screen's background. It can be any format supported by E3 on DrawPicture objects (*.bmp, *.gif, *.jpg, *.cur, *.ico, *.emf, *.wmf, *.png, and *.tif). Default value of this property is an empty **String**. This property is only valid if the **FillStyle** property is set to **12 - Picture**.

4.3.1.3.8 PicturePosition

Indicates the position of the selected picture on Screen's **PictureFile** property. This property is only valid if the **FillStyle** property is set to **12 - Picture**. Valid options for this property are described on the next table.

Available options for PicturePosition

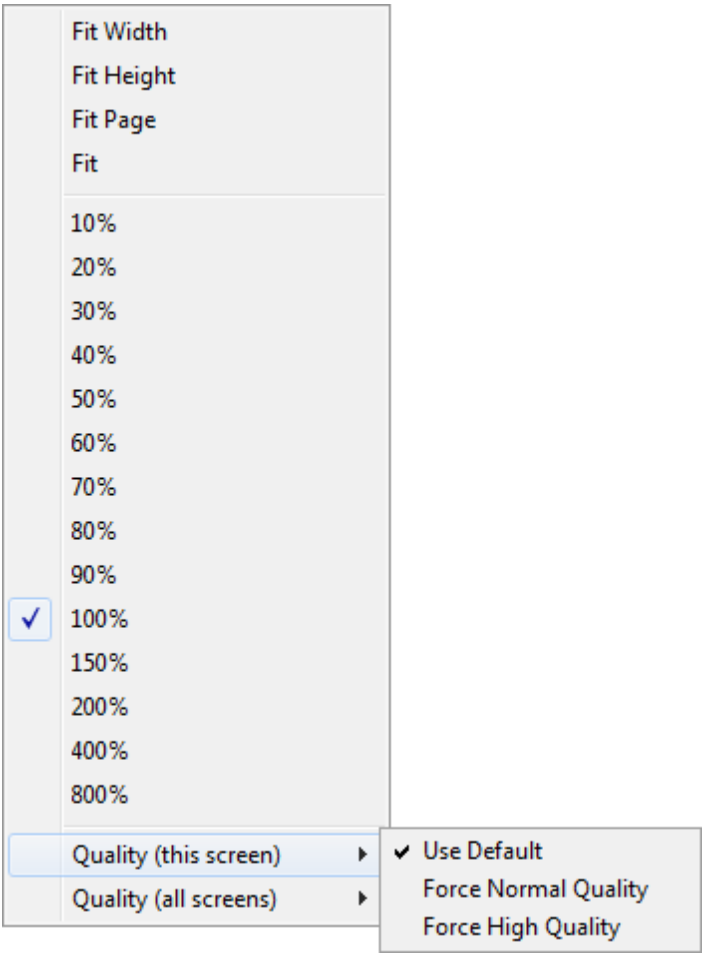
OPTION	DESCRIPTION
0 - Center	Picture with original size, centered on Screen
1 - Tile	Picture with original size, repeated as many times as needed to fill a Screen
2 - Stretch	Picture resized to fill a Screen
3 - TopLeft	Picture with original size, on the left upper corner of a Screen
4 - BottomLeft	Picture with original size, on the left lower corner of a Screen
5 - BottomRight	Picture with original size, on the right lower corner of a Screen
6 - TopRight	Picture with original size, on the right upper corner of a Screen

4.3.1.3.9 RenderQuality

Controls a Screen's drawing quality. Possible values for this property are the following:

- **0 - rqDefault**: This Screen's drawing quality uses the value defined on Viewer's **RenderQuality** property. This is the default value of this property, even in applications created on versions earlier than 4.0
- **1 - rqNormal**: Forces a normal quality mode (GDI) when drawing this Screen
- **2 - rqHighQuality**: Forces a high quality mode (GDI+) when drawing this Screen

The following figure displays a Screen's contextual menu at run time, with all respective configuration options for this property.



Screen's contextual menu at run time

4.3.2 Screen Objects

This section contains information about events, methods, and properties of Screen Objects.

4.3.2.1 Common Events

This section contains information about common events of all Screen Objects.

4.3.2.1.1 Click

Click()

This event occurs when the left mouse button is pressed on an object. This event does not occur if this object is not visible, or if its **Enabled** property is set to False. Visibility of an object depends on three factors: **Visible** property set to True, parent object visible, and object's **Layer** property present on a Screen layer.

4.3.2.1.2 DbClick

DbClick()

This event occurs when there is a double click, that is, the left mouse button is quickly pressed twice on an object. This event does not occur if this object is not visible, or if its **Enabled** property is set to False. Visibility of an object depends on three factors: **Visible** property set to True, parent object visible, and object's **Layer** property present on a Screen layer.

4.3.2.1.3 KeyDown

KeyDown(*KeyCode*, *Shift*)

This event occurs when a key is pressed and an object has the keyboard focus. Notice that this event is not generated if an object is not enabled (its **Enabled** property is set to False) or if this object does not have keyboard focus.

Variables of the KeyDown event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

NOTE: For a list with all key codes available for the *KeyCode* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.3.2.1.4 KeyUp

KeyUp(*KeyCode*, *Shift*)

This event occurs when a key is released and an object has keyboard focus. Notice that this event is not generated if this object is not enabled (its **Enabled** property is set to False) or if this object does not have keyboard focus.

Variables of the KeyUp event

NAME	DESCRIPTION
KeyCode	Integer identifying the ASCII character of the pressed key.
Shift	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys

NOTE: For a list with all key codes available for the *KeyCode* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.3.2.1.5 MouseDown

MouseDown(*Button*, *ShiftState*, *MouseX*, *MouseY*)

This event occurs when any mouse button is pressed on an object.

Variables of the MouseDown event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the key pressed along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the X coordinate where the mouse button was clicked.
MouseY	Displays the Y coordinate where the mouse button was clicked.

4.3.2.1.6 MouseUp

MouseUp(*Button*, *ShiftState*, *MouseX*, *MouseY*)

This event occurs when any previously clicked mouse button is released on an object. Use the **MouseUp** event to specify actions to perform only when releasing the mouse button.

Variables of the MouseUp event

NAME	DESCRIPTION
Button	Displays the pressed mouse button: • **1**: Left mouse button pressed • **2**: Right mouse button pressed
ShiftState	Displays the pressed key along with the mouse button: • **4**: SHIFT key • **8**: CTRL key • **12**: CTRL + SHIFT keys
MouseX	Displays the X coordinate where the mouse button was clicked.
MouseY	Displays the Y coordinate where the mouse button was clicked.

4.3.2.2 Common Methods

This section contains information about common methods of all Screen Objects.

4.3.2.2.1 BringToFront

BringToFront()

Places an object in front of all other Screen objects.

4.3.2.2.2 SendToBack

SendToBack()

Places an object at the back of all other Screen objects.

4.3.2.2.3 SetFocus

SetFocus()

Use the **SetFocus** method to move mouse or keyboard focus to a specific object.

4.3.2.3 Common Properties

This section contains all properties common to all Screen objects. Properties on this section do not apply to the following objects: **ActiveX (MSForms)**, **E3Chart**, **E3Browser**, and **E3Alarm**. These objects are treated later, on their specific chapters.

NOTE 1: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

NOTE 2: The following properties are common to all previously cited objects, including Groups of objects and Sliders.

4.3.2.3.1 Angle

The **Angle** property defines a rotation angle in degrees, counter clockwise, that an object should be rotated. This property also applies to children objects of this one, respecting rotation limitations of each child object. An object rotates according to its center, which can be edited during this rotation operation. Default value of this property is 0 (zero, no rotation).

4.3.2.3.2 BackgroundColor

This property specifies a color for an object's background. This color is used when the **BackgroundStyle** property is configured to 1 (one, opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties has a value different from 100. Another use of this color is when the **FillStyle** property is configured with values between 2 (two) and 8 (eight). This forces the remaining area to use that background color for filling. In scripts, use VBScript's **RGB** method to create a color associated to this property. Default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.3.3 BackgroundStyle

This property specifies a fill mode for an object's background. This property enables using the **VerticalPercentFill** and **HorizontalPercentFill** properties with values different from 100, and also set the **FillStyle** property with values between 2 (two) and 8 (eight). This forces the remaining area to use a background color configured in the **BackgroundColor** property for filling. The following table contains valid values for the **BackgroundStyle** property.

Available options for BackgroundStyle

OPTION	DESCRIPTION
0 - Transparent	No background is drawn
1 - Opaque	If visible, a background is drawn

4.3.2.3.4 BorderColor

Specifies a color for an object's border or line. This property is only used when the **BorderStyle** property is not configured to 5 (cinco, *Null*), where an object does not have a border. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is white (**RGB(255, 255, 255)**), except for Display and SetPoint objects, whose default value is dark gray (**RGB(128,128,128)**).

4.3.2.3.5 BorderStyle

The **BorderStyle** property determines a border style to apply to an object. The available options for this property are described on the next table.

Available options for BorderStyle

OPTION	DESCRIPTION
0 - Normal	Applies a solid border to an object (default)
1 - Dash	Applies a dashed border to an object
2 - Dot	Applies a dotted border to an object
3 - Dashdot	Applies a dash-dot border to an object
4 - Dashdotdot	Applies a dash-dot-dot border to an object
5 - Null	This object has no border

4.3.2.3.6 BorderWidth

Defines the width, in Himetric units, of an object's border or line. It is only used if the **BorderStyle** property is not configured to 5 (*Null*). Default value of this property is 0 (zero). This is an exception in E3's measuring system, because when its value is zero, line or border width is not defined in Himetric units, but in pixels. The **BorderWidth** property with a value equal to 0 (zero) indicates a one-pixel width.

4.3.2.3.7 Effect3D

Applies a 3D effect in the selected object. The available options for this property are described on the following table.

Available options for Effect3D

OPTION	DESCRIPTION
0 - No3D	Transparent (default)
1 - Raised	Applies a raised 3D effect
2 - Sunken	Applies a sunken 3D effect

4.3.2.3.8 Effect3D_X

Specifies the dimension of a 3D effect on an object's horizontal axis (X axis). Default value of this property is 30.

4.3.2.3.9 Effect3D_Y

Specifies the dimension of a 3D effect on an object's vertical axis (Y axis). Default value of this property is 30.

4.3.2.3.10 Effect3DColorBase

Determines a color for an object's 3D base effect. Default value of this property is black (**RGB(0, 0, 0)**).

4.3.2.3.11 Effect3DColorTop

Determines a color for an object's 3D top effect. Default value of this property is white (**RGB(255, 255, 255)**).

4.3.2.3.12 Enabled

Enables or disables an object, that is, focus and response to user-generated events. If this property is set to True, an object can receive focus, respond to user-generated events, and it is accessible via scripts (default). Otherwise, users cannot interact with this object by using the mouse pointer, by pressing keys or shortcut keys, and this object appears faded. In addition, if this object displays a bitmap. that bitmap remains faded whenever this object is disabled.

The **Enabled** and **Locked** properties (available in **Check Box**, **Option Button**, **Combo**, **Command Button**, **List**, **Toggle Button**, and **Text** objects) are interconnected. When the **Enabled** and **Locked** properties are both configured as True, an object can receive focus and appear normally on Screen, and its data can be copied but not edited. When **Enabled** is equal to True, but **Locked** is equal to False, data can be copied and also edited. However, when **Enabled** is equal to False, an object does not receive focus and remains faded on Screen, regardless of **Locked** status. In addition, data cannot be copied nor edited.

Users can combine configurations of **Enabled** and **TabStop** properties to prevent users from selecting a Command Button with the TAB key, although users are still allowed to click that button. Defining the **TabStop** property as False means that a Command Button does not appear on the tab order. However, if **Enabled** is equal to True, users can still click that Command Button, as long as the **TakeFocusOnClick** property is defined as True.

4.3.2.3.13 FillStyle

This property specifies an object's fill style. The following table contains valid values for the **FillStyle** property.

Available options for FillStyle

OPTION	DESCRIPTION
0 - Solid	Fill is solid (default)
1 - Hollow	No fill
2 - Horizontal	Fill has horizontal stripes
3 - Vertical	Fill has vertical stripes

OPTION	DESCRIPTION
4 - Downward	Fill has stripes down from left to right in 45 degrees
5 - Upward	Fill has stripes up from left to right in 45 degrees
6 - Cross	Fill has horizontal and vertical stripes
7 - DiagonalCross	Fill has stripes up and down from left to right in 45 degrees
8 - Gradient	Fill has a gradient using both the **ForegroundColor** and the **BackgroundColor** property. This effect is determined by the **GradientStyle** property
9 - SemiTransparent	The object remains semitransparent
10 - MouseArea	No fill, but this object remains responsive to events

NOTE: The **FillStyle** property is not available for **Picture**, **Dynamic Move**, and **Dynamic Rotate** objects.

4.3.2.3.14 ForegroundColor

This property specifies a color for an object's foreground. This color is used when the **FillStyle** property is set to 0 (zero, solid) or between 2 (two) and 9 (nine). In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is blue (**RGB(0, 0, 255)**), except for Display and SetPoint objects, whose default value is white (**RGB(255, 255, 255)**).

4.3.2.3.15 Frame

Returns an object's parent Frame. This property is only accessible at run time.

4.3.2.3.16 GradientStyle

This property specifies a fill style for an object's gradient. This property is only used when the value of the **FillStyle** property is set to 8 (eight, **Gradient**). Gradients consider its change as starting from the color configured in the **ForegroundColor** property and moving towards the color configured in the **BackgroundColor** property.

Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient from left to right
1 - RightToLeft	Vertical gradient from right to left
2 - VerFromCenter	Vertical gradient from center to border
3 - VerToCenter	Vertical gradient from border to center
4 - BottomUp	Horizontal gradient from bottom to top
5 - TopDown	Horizontal gradient from top to bottom

OPTION	DESCRIPTION
6 - HorzFromCenter	Gradient from center to border
7 - HorzToCenter	Gradient from border to center
8 - DiagUpRight	Diagonal gradient to top with foreground color on the right (default)
9 - DiagUpLeft	Diagonal gradient to top with foreground color on the left
10 - DiagUpFromCenter	Diagonal gradient to top from center to border
11 - DiagUpToCenter	Diagonal gradient to top from border to center
12 - DiagDownLeft	Diagonal gradient to bottom with foreground color on the left
13 - DiagDownRight	Diagonal gradient to bottom with foreground color on the right
14 - DiagDownFromCenter	Diagonal gradient to bottom from center to border
15 - DiagDownToCenter	Diagonal gradient to bottom from border to center
16 - SpotSouthEast	Gradient with foreground color from lower right corner
17 - SpotSouthWest	Gradient with foreground color from lower left corner
18 - SpotNorthWest	Gradient with foreground color from upper left corner
19 - SpotNorthEast	Gradient with foreground color from upper right corner
20 - SpotFromCenter	Gradient with background color from center to border
21 - SpotToCenter	Gradient with background color from border to center

IMPORTANT: A large amount of objects simultaneously displayed with a gradient may lead to a poor performance when updating a Screen. Using images may solve this problem.

4.3.2.3.17 HasFocus

◩ This property determines that the selected object has the focus. This property is only accessible at run time.

4.3.2.3.18 Height

9 Determines an object's height.

4.3.2.3.19 HorizontalFillStyle

Defines an object's horizontal fill. This property works along with the **HorizontalPercentFill** property, which informs the percentage of an object must be filled. These two properties allows a simulation of a level fill in an object, such as a tank level, for example.

Available options for HorizontalFillStyle

OPTION	DESCRIPTION
0 - FillLeftToRight	Fill percentage is from left to right (default)
1 - FillRightToLeft	Fill percentage is from right to left
2 - FillCenterToEdgesH	Fill percentage is from center to border

4.3.2.3.20 HorizontalPercentFill

Use the **HorizontalPercentFill** property to specify a percentage of an object's horizontal area to fill. Acceptable values for this property vary from 0 (zero) to 100. This property works along with the **HorizontalFillStyle** property, which informs how that fill is performed. Default value of this property is 100.

4.3.2.3.21 Layer

This property defines in which layers an object must appear. This value represents a 32-bit mask, one bit for each layer. Therefore, up to 32 individual layers can be defined. Thus, objects can be logically grouped and displayed or hidden only by changing the **Layer** property's mask.

4.3.2.3.22 MouseOver

The **MouseOver** property informs whether the mouse pointer is on a Screen. If True, the **MouseOver** property is enabled. Otherwise, it is disabled. This is a read-only property and it is only available at run time. Default value of this property is False.

4.3.2.3.23 MouseOverChild

The **MouseOverChild** property informs whether the mouse pointer is on one of the objects inserted on a Screen. If True, the **MouseOverChild** property is enabled. Otherwise, it is disabled. This is a read-only property and it is only available at run time. Default value of this property is False.

4.3.2.3.24 Screen

Returns an object's parent Screen. This property is only accessible at run time.

4.3.2.3.25 Shadow

Indicates the presence of a shadow effect on an object. If configured as True, this object has a shadow, whose coordinates are established by the **ShadowX** and **ShadowY** properties. Otherwise, this object does not have a shadow effect. Default value of this property is False.

4.3.2.3.26 ShadowColor

Specifies a fill color for an object's shadow. This color is used when the **Shadow** property is set to True. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is dark gray (**RGB(128, 128, 128)**).

4.3.2.3.27 ShadowX

Defines the left vertical coordinate of an object's shadow, in Himetric units. This shadow is always relative to the object's **X** property. Positive values indicate that this shadow is on the right side of an object, and negative ones on the left side. Default value of this property is 200.

4.3.2.3.28 ShadowY

Defines the upper horizontal coordinate of an object's shadow, in Himetric units. This shadow is always relative to the object's **Y** property. Positive values indicate that this shadow is below an object, and negative ones above. Default value of this property is 200.

4.3.2.3.29 TabStop

This property determines how TAB key is used on an application. If this property is set to True, users can use the TAB key. Otherwise, this key cannot be used.

4.3.2.3.30 Tip

The **Tip** property displays a popup text when the mouse pointer is briefly over an object at run time.

4.3.2.3.31 VerticalFillStyle

Defines an object's vertical fill. This property works along with the **VerticalPercentFill** property, which informs the percentage of an object to fill. These two properties allow a simulation of a level fill on an object.

Available options for VerticalFillStyle

OPTION	DESCRIPTION
0 - FillBottomToTop	Fill percentage is from bottom to top

OPTION	DESCRIPTION
1 - FillTopToBottom	Fill percentage is from top to bottom
2 - FillCenterToEdgesV	Fill percentage is from center to border

4.3.2.3.32 VerticalPercentFill

Use the **VerticalPercentFill** property to specify a percentage of an object's vertical area to fill. Acceptable values for this property vary from 0 (zero) to 100. This property works along with the **VerticalFillStyle** property, which indicates how this fill occurs. Default value of this property is 100.

4.3.2.3.33 Visible

This property defines an object's visibility. If configured as True, this object remains visible, as long as the following factors are also present: this object's parent object must also be visible and this object's **Layer** property must also be available on a Screen layer.

4.3.2.3.34 Width

Determines an object's width, in Himetric units.

4.3.2.3.35 X

The **X** property defines an object's left horizontal coordinate, in Himetric units.

4.3.2.3.36 Y

The **Y** property defines an object's upper vertical coordinate, in Himetric units.

4.3.2.4 Group

This section contains information about **properties** of the Group of objects. This object does not have events nor methods associated to it.

4.3.2.4.1 Properties

This section contains information about the properties of the Group of objects.

4.3.2.4.1.1 EnableOverrideLineColor

This property enables or disables a Group object to override original colors of lines on objects contained in that Group. If the **EnableOverrideLineColor** is enabled, original colors of lines on objects in that Group are replaced by the color defined in the **OverrideLineColor** property. Otherwise, each object contained in that Group displays its original line color. Default value of this property is False.

4.3.2.4.1.2 OverrideFillColor

When the **OverrideFillMode** property is set to 2 (two) or 3 (three), the **OverrideFillColor** property is used to define a color to use when filling adjacent objects in a Group, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.4.1.3 OverrideFillMode

The **OverrideFillMode** property specifies a fill mode for objects in a Group. It changes the original image's fill mode without changing original object's fill settings.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Object's original fill.
1 - WireFrame	Objects are not filled, they draw only their borders.
2 - SolidFill	Filling of objects in a Group is solid with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	Filling of objects in a Group is solid with the color specified in the **OverrideFillColor** property, but it takes into account the intensity of the original fill color for each object.

4.3.2.4.1.4 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define a color to use on object's line of a Group, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.5 Round Rectangle

This section contains information about the **properties** of the Rounded Rectangle object. This object does not have events nor methods associated to it.

4.3.2.5.1 Properties

This section contains information about the properties of the Round Rectangle object.

4.3.2.5.1.1 RoundAspectX

The **RoundAspectX** property defines the dimension of rectangle sizes on X axis. Thus, according to the value defined on this property, rectangle corners change their shape, from a rectangle to an ellipse. This property may vary its value from 0,1 to 1,0. Example:

```
Sub RoundRect_Click()
  RoundAspectX = 0.5
End Sub
```

4.3.2.5.1.2 RoundAspectY

The **RoundAspectY** property defines the dimension of rectangle height on Y axis. Thus, according to the value defined on this property, rectangle corners change their shape, from a rectangle to an ellipse. This property may vary its value from 0,1 to 1,0. Example:

```
Sub RoundRect_Click()
  RoundAspectY = 0.5
End Sub
```

4.3.2.6 Arc of Ellipse

This section contains information about the **properties** of the Arc of Ellipse object. This object does not have events nor methods associated to it.

4.3.2.6.1 Properties

This section contains information about the properties of the Arc of Ellipse object.

4.3.2.6.1.1 ArcBeginAngle

This property configures an initial angle for this object's arc, in degrees. The accepted interval for this property ranges from 0 (zero) to 359. Arc's style and shape also depend on the **ArcEndAngle** and **ArcStyle** properties. Default value of this property is 0 (zero). Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcBeginAngle = 12
End Sub
```

4.3.2.6.1.2 ArcEndAngle

This property configures a final angle for this object's arc, in degrees. The accepted interval for this property ranges from 0 (zero) to 359. Arc's style and shape also depend on the **ArcBeginAngle** and **ArcStyle** properties. Default value of this property is 270. Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcEndAngle = 12
```

```
End Sub
```

4.3.2.6.1.3 ArcStyle

This property specifies a border or line style for this object. This border is drawn according to a defined style, using a color specified by **BorderColor** and a width specified by **BorderWidth**. The next table contains valid values for the **ArcStyle** property.

Available options for ArcStyle

OPTION	DESCRIPTION
0 - arc	Drawing style is an arc
1 - chord	Drawing style is a chord, connecting a starting and an ending point
2 - pie	Drawing style is a pie (default)

Example:

```
Sub CommandButton9_Click()
  Screen.Item("Arc1").ArcStyle = 1
End Sub
```

4.3.2.7 Picture

This section contains information about the **properties** of the Picture object. This object does not have events nor methods associated to it.

4.3.2.7.1 Properties

This section contains information about the properties of the Picture object.

4.3.2.7.1.1 BackgroundColor

This property specifies a fill color for an object's background. This color is used when the **BackgroundStyle** property is set to 1 (one, opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties is different from 100. In scripts, use VBScript's **RGB** method to build a color to link to this property. Default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.7.1.2 BackgroundStyle

This property specifies a fill mode for this object's background. This property enables using the **VerticalPercentFill** and **HorizontalPercentFill** properties with values different from 100, allowing that the remaining area uses the color configured in the **BackgroundColor** property as its fill. The next table contains valid values for the **BackgroundStyle** property.

Available options for BackgroundStyle

OPTION	DESCRIPTION
0 - Transparent	No object background is drawn
1 - Opaque	If visible, background is drawn

4.3.2.7.1.3 Convert

This property allows converting a picture. If this property is set to 0 (zero), users can view that conversion. Otherwise, it is not possible to preview that conversion. This property only accepts values 0 (zero) and 1 (one). Default value of this property is 0 (zero).

4.3.2.7.1.4 EnableOverrideLineColor

This property enables or disables an object to overwrite the original color of a picture's line by the color defined in the **OverrideLineColor** property. If the **EnableOverrideLineColor** property is enabled, it allows modifying the original color of an object's line by the color of the **OverrideLineColor** property. Otherwise, a Picture object displays its original color.

4.3.2.7.1.5 Filename

Defines a picture's file name linked to this object. This file's path can be a full file path on disk, as well as a path relative to an application (when this picture file is inserted as an application Resource). Default value of this property is empty. The next table contains all supported file types.

Supported picture file types

FILE	FILTER	DESCRIPTION	FILTER
Bitmap	BMP	No	Yes
Graphics Interchange Format	GIF	No	Yes
Joint Photographic Experts Group	JPG	No	Yes
ICO File Format	ICO	No	Yes
Windows Metafile	WMF	No	Yes
Enhanced Metafile	EMF	No	Yes
Portable Network Graphics	PNG	No	Yes
Tagged Image File Format	TIF	No	Yes

4.3.2.7.1.6 HorizontalFillStyle

Defines a horizontal fill for this object. This property works along with the **HorizontalPercentFill**, property, which informs a percentage of this object that must be filled. These two properties allow simulating a level fill in an object, such as a tank level, for example.

Available options for HorizontalFillStyle

OPTION	DESCRIPTION
0 - FillLeftToRight	Fill percentage is from left to right (default)
1 - FillRightToLeft	Fill percentage is from right to left
2 - FillCenterToEdgesH	Fill percentage is from center to border

4.3.2.7.1.7 HorizontalPercentFill

Use the **HorizontalPercentFill** property to specify a percentage of an object's horizontal area to fill. Acceptable values for this property range from 0 (zero) to 100. This property works along with the **HorizontalFillStyle** property, which informs how this fill occurs. Default value of this property is 100. Example:

```
Sub Circle1_OnStartRunning()
  HorizontalPercentFill = 200
End Sub
```

4.3.2.7.1.8 OverrideFillColor

When the **OverrideFillMode** property is set to 2 (two) or 3 (three), the **OverrideFillColor** property is used to define a color to fill a picture, instead of its original color. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawPicture1_Click()
  ' When clicking this object sets the Override
  ' mode to solid and changes this picture's fill
  ' color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property is only effective when a Picture object is working with Metafiles (WMF or EMF).

4.3.2.7.1.9 OverrideFillMode

The **OverrideFillMode** property specifies a fill mode for this object's picture, when displaying a Windows Metafile. It changes the picture's original fill mode without changing the file defined in the **Filename** property. The next table contains

valid values for the **OverrideFillMode** property.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Picture keeps its original fill (default).
1 - WideFrame	Picture is not filled.
2 - SolidFill	Picture is filled with the color specified by the **OverrideFillColor** property.
3 - ByBrightness	Picture is filled with the color specified by the **OverrideFillColor** property, but considers the intensity of picture's original color.

Example:

```
Sub DrawPicture1_Click()
  ' When clicking this object sets the
  ' Override mode to solid
  ' and changes picture's fill
  ' color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property is only effective when a Picture object is working with Metafiles (WMF or EMF).

4.3.2.7.1.10 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define a color to use as a picture's line, instead of its original color. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawPicture1_Click()
  OverrideLineColor = RGB(0, 0, 255)
End Sub
```

NOTE: This property is only effective when a Picture object is working with Metafiles (WMF or EMF).

4.3.2.7.1.11 Shadow

Indicates the presence of a shadow effect in this object. If set to True, this object has a shadow, whose coordinates are established by the **ShadowX** and **ShadowY** properties. Otherwise, this object does not have a shadow effect. Default value of this property is False.

4.3.2.7.1.12 ShadowColor

Specifies a fill color for this object's shadow. This color is used when the **Shadow** property is set to True. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is light gray (**RGB(128, 128, 128)**). Example:

```
Sub Button1_Click()
  ' Changes the background color of this button
  ' to light gray when clicking it
  ShadowColor = RGB(192, 192, 192)
End Sub
```

NOTE: This property is only effective when a Picture object is working with Metafiles (WMF or EMF).

4.3.2.7.1.13 ShadowX

Defines a vertical coordinate to the left of this object's shadow, in Himetric units. This shadow is always relative to this object's **X** property. Positive values indicate that this shadow is on object's right side, and negative values on object's left side. Default value of this property is 200.

4.3.2.7.1.14 ShadowY

Defines an upper horizontal coordinate for this object's shadow, in Himetric units. This shadow is always relative to this object's **Y** property. Positive values indicate that this shadow is under the object, and negative values that it is above the object. Default value of this property is 200.

NOTE: This property is only effective when a Picture object is working with Metafiles (WMF or EMF).

4.3.2.7.1.15 TransparentColor

When the **TransparentMode** property is set to 1 (one), this property defines which color on this picture is not drawn, and that picture remains transparent in these areas. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub DrawPicture1_Click()
  ' Leaves the blue color transparent
  ' when clicking this object
  TransparentMode = 1 ' ByColor
  TransparentColor = RGB(0, 0, 255)
End Sub
```

4.3.2.7.1.16 TransparentMode

The **TransparentMode** property specifies a transparency effect mode for this picture. The next table contains possible values for this property.

Available options for TransparentMode

OPTION	DESCRIPTION
0 - Disabled	No transparency is performed
1 - ByColor	Transparency uses a color defined in the **TransparentColor** property
2 - ByPercent	Picture remains transparent with a transparency percentage specified in the **TransparentPercent** property

Example:

```
Sub DrawPicture1_Click()
  ' Leaves the blue color transparent
  ' when clicking this object
  TransparentMode = 1 ' ByColor
  TransparentColor = RGB(0, 0, 255)
End Sub
```

4.3.2.7.1.17 TransparentPercent

When the **TransparentMode** property is set to 2 (two), this property defines how transparent a picture is displayed, ranging from 0 (zero, fully transparent) to 100 (opaque or solid). Example:

```
Sub DrawPicture1_Click()
  ' Leaves this picture transparent
  ' when clicking this object
  TransparentMode = 2 ' ByPercent
  TransparentPercent = 50 ' 50% transparent
End Sub
```

4.3.2.7.1.18 VerticalFillStyle

Defines a vertical fill for this object. This property works along with the **VerticalPercentFill** property, which informs a percentage of this object to fill. These two properties allow simulating a level fill in an object.

Available options for VerticalFillStyle

OPTION	DESCRIPTION
0 - FillBottomToTop	Fill percentage is from bottom to top
1 - FillTopToBottom	Fill percentage is from top to bottom
2 - FillCenterToEdgesV	Fill percentage is from center to border

4.3.2.7.1.19 VerticalPercentFill

9 Use the **VerticalPercentFill** property to specify a percentage of this object's vertical area to fill. Acceptable values for this property range from 0 (zero) to 100. This property works along with the **VerticalFillStyle** property, which informs how this fill occurs. Default value of this property is 100.

4.3.2.8 Text, Display, and SetPoint

This section contains information about **events** and **properties** of Text, Display, and SetPoint objects. These objects do not have methods associated to them.

4.3.2.8.1 Events

This section contains information about the events of Text, Display, and SetPoint objects.

4.3.2.8.1.1 Validate

Validate(*Cancel*, *NewValue*)

Occurs after testing SetPoint limits (please check **MinLimit**, **MaxLimit**, and **EnableLimits** properties) and before sending a SetPoint value to a Tag. The purpose of this event is allowing users to cancel sending SetPoint's value to a Tag.

The *Cancel* parameter is a **Boolean** that indicates if attributing a SetPoint's value to a Tag must be canceled (*Cancel* equal to True). Default is False, that is, SetPoint's value is sent to a Tag. *NewValue* is the evaluated value. The old value can be accessed in SetPoint's **Value** property. Example:

```
Sub Text1_Validate(Cancel, NewValue)
  ' Displays a MessageBox that asks a user
  ' if the new SetPoint value should be used
  message = "Current value: " & value & vbnewline & _
    "New value: " & NewValue & vbnewline & vbnewline & _
    "Do you accept the new value?"
  If MsgBox (message, vbQuestion + vbYesNo, _
    "Validate Event") = vbNo Then
    Cancel = True
  End If
End Sub
```

4.3.2.8.2 Properties

This section contains information about the properties of Text, Display, and SetPoint objects.

4.3.2.8.2.1 EnableLimits

Indicates if there is a check on text limits. When **EnableLimits** is equal to True, and users insert a non-number value or a value outside the limits defined in the **MinLimit** and **MaxLimit** properties, an error message is displayed (the **IsSetPoint** property must be equal to True). Example:

```
Sub CommandButton1_Click()
  Screen.Item("Text1").EnableLimits = _
    Not(Screen.Item("Text1").EnableLimits)
End Sub
```

4.3.2.8.2.2 Format

The Format property specifies a format type to attribute to this object. It allows changing the way data is displayed without changing its value. This property can be edited manually or configured using a format window. Its usage is similar to formats used on spreadsheets, following the same syntax. The next table contains all supported data types.

Data types supported by Format

DATA TYPE	DESCRIPTION
Number	Decimal, scientific, hexadecimal, binary, or octal output
Text	General text
Boolean	**Boolean** values
Date/Time	Gregorian calendar

4.3.2.8.2.3 IsSetPoint

The **IsSetPoint** property is used to determine whether an object behaves as a SetPoint, that is, if it allows editing its **Value** property. The **Value** property is a **Variant**, which may assume any and every type of value. The **IsSetPoint** property is a **Boolean** and assumes the following behavior: if it is equal to True, it allows edition and, if it is False, it does not allow edition. This behavior can be viewed when Viewer is executing. Default value is False.

4.3.2.8.2.4 KeepFormatWhenEditing

Allows an object's value to be edited with or without format. The available options are the following:

- **0 - kfNever**: Value is always edited without format (default)
- **1 - kfAutomatic**: Allows a value to be edited in the formatted value, if E3 detects that the formatted text can be interpreted as a value

If a format is considered as incompatible, object's value is edited without format.

4.3.2.8.2.5 MaxLimit

Contains the maximum allowed value for this object (the **EnableLimits** property must be equal to True). Example:

```
Sub CommandButton3_Click()
  Screen.Item("Text1").MaxLimit = Screen.Item("Text6").Value
End Sub
```

4.3.2.8.2.6 MinLimit

Contains the minimum allowed value for this object (the **EnableLimits** property must be equal to True). Example:

```
Sub CommandButton2_Click()
  Screen.Item("Text1").MinLimit = Screen.Item("Text5").Value
End Sub
```

4.3.2.8.2.7 Multiline

The Multiline property indicates whether a Text contains multiple lines (True) or it is a simple text box (False). This behavior can be viewed when Viewer is executing. Default value is False.

4.3.2.8.2.8 SetPointDataType

Determines a type of value that is sent from a SetPoint to a Tag. Possible values for this property are described on the next table.

Available options for SetPointDataType

OPTION	DESCRIPTION
0 - stCurrentType	Maintains current value type in a SetPoint (please check next)
1 - stChar	8-bit signed integer value
2 - stByte	8-bit unsigned integer value
3 - stWord	16-bit unsigned integer value
4 - stInteger	16-bit signed integer value
5 - stLong	32-bit signed integer value
6 - stDWord	16-bit unsigned integer value
7 - stSingle	32-bit floating point value
8 - stDouble	64-bit floating point value
9 - stDateTime	Date and time value
10 - stString	Text

When a typed text is sent by a SetPoint, it first tries to convert that value to the

configured type (**Word**, **String**, **Double**, etc.). If this conversion is not possible, that is, the typed value is not valid for the selected type, no value is sent (for example, if users type "-1", and type is **Byte**). When this property's value is **0 - stCurrentType**, a data type sent by this SetPoint comes from the previous value available in that object. If that previous value is *Empty* or *Null*, no conversion is performed and the typed value is sent as text. Example:

```
Sub Combobox1_Change()
  Screen.Item("Text1").SetPointDataType = CInt(Left(Value, 2))
End Sub
```

4.3.2.8.2.9 StretchText

Resizes an object's type. When enabling the **StretchText** property, this object automatically resizes its text font size, so that the area occupied by it always remains the same. Otherwise, if the **StretchText** property is set to False, no resizing is performed.

4.3.2.8.2.10 TextAlignment

This property specifies a horizontal alignment for a text displayed in this object. Possible values for this property are described on the next table.

Available options for TextAlignment

OPTION	DESCRIPTION
0 - LeftAlignment	Text's horizontal alignment is on the left
1 - CenterAlignment	Text's horizontal alignment is centered
2 - RightAlignment	Text's horizontal alignment is on the right

4.3.2.8.2.11 TextColor

This property specifies a color for a text font to display. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value is black (**RGB(0, 0, 0)**).

4.3.2.8.2.12 TextFont

Defines information about a font used in this object. This property cannot be used in Links and contains all sub-properties described on the next table, which can be changed via script.

TextFont sub-properties

NAME	TYPE	DESCRIPTION
Name	String	Font name.
Size	Floating Point	Font size, in points.

NAME	TYPE	DESCRIPTION
Bold	Boolean	Indicates whether text has a bold style.
Italic	Boolean	Indicates whether text has an italic style.
Underline	Boolean	Indicates whether text has an underline style.
Strikethrough	Boolean	Indicates whether text has a strike-through style.
Weight	Integer	Indicates a variation on font's bold effect (the **Bold** sub-property). Values for this sub-property may range from 0 (zero, does not interfere with the bold effect) to 1000.
Charset	Integer	Font character set. Please check the next table of character sets for a list of all available values for this sub-property.

Available values for Charset sub-property

VALUE	DESCRIPTION
0	ANSI (*American National Standards Institute*)
1	Default (displays the font's available character set)
2	Symbol
128	Japanese (Shift-JIS)
129	Korean
134	Chinese Simplified (GBK)
136	Chinese Traditional (Big5)
161	Greek
162	Turkish
177	Hebrew
178	Arabic
186	Baltic
204	Russian
222	Thai
238	Central Europe
255	OEM (*Original Equipment Manufacturer*)

4.3.2.8.2.13 Value

The **Value** property is a **Variant**, which may assume any and every type of value displayed by an object. Usually this property contains a text, because it is automatically filled in when creating a new Text object. The **IsSetPoint** property is used to determine whether a Text object has the same behavior of a SetPoint, that is, allows editing its **Value** property. Example:

```
Sub DrawString1_OnStartRunning()
  ' Reads a value from a Tag and displays its Text
  Dim obj
  Set obj = Application.GetObject("DataServer1.DemoTag1")
  Value = "DemoTag1 value = " & obj.Value
End Sub
```

4.3.2.8.2.14 VertTextAlignment

Determines a vertical alignment for an object's text. Possible values for this property are described on the next table.

Available options for VertTextAlignment

OPTION	DESCRIPTION
0 - TopAlignment	Text's vertical alignment is on top of this object
1 - MidAlignment	Text's vertical alignment is centered in this object
2 - BottomAlignment	Text's vertical alignment is at the bottom of this object

4.3.2.8.2.15 WordWrap

Enables or disables a line break in this text, if the available area for this text overrides the limits defined by an object. For this property to work, the **Multiline** property must be set to True.

4.3.2.9 Scale

This section contains information about **properties** of the Scale object. This object does not have events nor methods associated to it.

4.3.2.9.1 Properties

This section contains information about the properties of the Scale object.

4.3.2.9.1.1 BackgroundColor

This property specifies a fill color for this object's background. This color is used when the **BackgroundStyle** property is set to 1 (one, opaque) and one of the **VerticalPercentFill** or **HorizontalPercentFill** properties has a value different from 100. Another usage of this color is when the **FillStyle** property is set to values between 2 (two) and 8 (eight). This allows the remaining area to use that background color for filling. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.9.1.2 BorderColor

This property determines a border color to apply to a Scale object. With this property, users can apply a default color or customize that color by editing it. Default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub Scale1_Click()
  BorderColor = RGB (255, 0, 0)
End Sub
```

4.3.2.9.1.3 BorderStyle

The **BorderStyle** property determines a border style applied to a Scale object. Possible values for this property are described on the next table.

Available options for BorderStyle

OPTION	DESCRIPTION
0 - Normal	Applies a solid border on Scale's vertical grid (default)
1 - Dash	Applies a dashed-line border on this Scale
2 - Dot	Applies a dotted border on this Scale
3 - Dashdot	Applies a dash-dot border on this Scale
4 - Dashdotdot	Applies a dash-dot-dot border on this Scale
5 - Null	This object has no border

4.3.2.9.1.4 BorderWidth

This property determines a border width (in pixels) of a Scale object. By using this property, users can configure a border width, without changing its structure. Default value of this property is 0 (zero).

4.3.2.9.1.5 ForegroundColor

This property specifies a fill color for an object's foreground. This color is used when the **FillStyle** property is set to 0 (zero, solid) or between 2 (two) and 9 (nine). In

scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is blue (**RGB(0, 0, 255)**). Example:

```
Sub Button1_Click()
  ' Changes button's background color
  ' to green when clicking it
  ForegroundColor = RGB(0, 255, 0)
End Sub
```

4.3.2.9.1.6 Format

The **Format** property specifies a format type to attribute to this object. It allows changing the way data is displayed without changing their values. This property can be edited manually or configured using a format window. Its usage is similar to formats used on spreadsheets, following the same syntax. The next table contains all supported data types.

Data types supported by Format

DATATYPE	DESCRIPTION
Number	Decimal, scientific, hexadecimal, binary, or octal output
Text	General text
Boolean	**Boolean** values
Date/Time	Gregorian calendar

4.3.2.9.1.7 GradientStyle

This property specifies a fill style for an object's gradient. This property is only used when the **FillStyle** property is set to 8 (eight, **Gradient**). Gradients consider a color change ranging from **ForegroundColor** to **BackgroundColor**.

Available options for GradientStyle

OPTION	DESCRIPTION
0 - LeftToRight	Vertical gradient from left to right
1 - RightToLeft	Vertical gradient from right to left
2 - VerFromCenter	Vertical gradient from center to border
3 - VerToCenter	Vertical gradient from border to center
4 - BottomUp	Horizontal gradient from bottom to top
5 - TopDown	Horizontal gradient from top to bottom
6 - HorzFromCenter	Gradient from center to border
7 - HorzToCenter	Gradient from border to center
8 - DiagUpRight	Diagonal top gradient with foreground color on the right (default)
9 - DiagUpLeft	Diagonal top gradient with foreground color on the left
10 - DiagUpFromCenter	Diagonal top gradient from center to border

OPTION	DESCRIPTION
11 - DiagUpToCenter	Diagonal top gradient from border to center
12 - DiagDownLeft	Diagonal bottom gradient with foreground color on the left
13 - DiagDownRight	Diagonal bottom gradient with foreground color on the right
14 - DiagDownFromCenter	Diagonal bottom gradient from center to border
15 - DiagDownToCenter	Diagonal bottom gradient from border to center
16 - SpotSouthEast	Gradient with foreground color from right lower corner
17 - SpotSouthWest	Gradient with foreground color from left lower corner
18 - SpotNorthWest	Gradient with foreground color from left upper corner
19 - SpotNorthEast	Gradient with foreground color from right upper corner
20 - SpotFromCenter	Gradient with background color from center to border
21 - SpotToCenter	Gradient with background color from border to center

IMPORTANT: A large number of objects displayed simultaneously with gradients leads to poor performance when updating a Screen. Using pictures may solve this problem.

Example:

```
Sub Button1_Click()
  ' Objeto gets a gradient
  FillStyle = 8 ' GradientFill
  GradientStyle = 0 ' LeftToRight
End Sub
```

4.3.2.9.1.8 LineColor

Determines a line color for Scale's ticks and minor ticks. To determine a legend color with object's numbers, use the **TextColor** property. Default value for this property is black (**RGB(0, 0, 0)**).

4.3.2.9.1.9 MaximumValue

9 This property determines the maximum value reached by a Scale. Default value of this property is 100. Example:

```
Sub CommandButton_Click()
  ' When clicking this button, opens a MessageBox
```

```
  ' indicating the maximum value for this property
  MsgBox CSTr(Screen.Item("Scale1").MaximumValue)
End Sub
```

4.3.2.9.1.10 MinimumValue

This property determines the minimum value reached by a Scale. Default value of this property is 0 (zero). Example:

```
Sub CommandButton1_Click()
  ' When clicking this button, opens a MessageBox
  ' indicating the minimum value for this property
  MsgBox _
    CSTr(Application.GetObject("Data.Scale1").MinimumValue)
End Sub
```

4.3.2.9.1.11 MinorTicks

This property determines the amount of minor ticks on a Scale. Default value of this property is 3 (three). Example:

```
Sub CommandButton1_Click()
  ' Displays the total amount
  ' of minor ticks on a scale
  MsgBox CStr(Screen.Item("Scale1").MinorTicks)
End Sub
```

4.3.2.9.1.12 MinorTicksPercentSize

This property determines the size of minor ticks subdividing each Scale's measurement. Default value of this property is 10.

4.3.2.9.1.13 ScaleAlignment

This property determines a type of alignment for a Scale:

- **0 - RightSide**: To the right (default value)
- **1 - LeftSide**: To the left

4.3.2.9.1.14 ShowText

This property determines text visibility on a Scale's legend. If this property is set to True, text is displayed. Otherwise, this object only displays Scale lines and minor ticks. Default value of this property is True.

4.3.2.9.1.15 StretchText

This property determines the application of a stretch to a Scale's text (if Scale's width or height change, text follows that change). If this property is enabled, this object follows width and height variations as it is transformed. Otherwise, text

maintains its original settings. Default value is of this property is False.

4.3.2.9.1.16 TextAlignment

The **TextAlignment** property determines this object's text alignment. Possible values for this property are described on the next table.

Available option for TextAlignment

OPTION	DESCRIPTION
0 - leftAlignment	Text alignment is on the left (default)
1 - centerAlignment	Text alignment is centered
2 - rightAlignment	Text alignment is on the right

4.3.2.9.1.17 TextColor

Determines a font color applied on a Scale's numbered legend. To determine a line color with Scale's ticks and minor ticks, use the **LineColor** property. Default value of this property is black (**RGB(0, 0, 0)**).

4.3.2.9.1.18 TextFont

A The **TextFont** property determines a font to apply to a Scale. The selected font is applied to the whole Scale. This property cannot be used in Links. Please check the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

4.3.2.9.1.19 Ticks

9 Determines the number of ticks on a Scale. Default value of this property is 5 (five).

4.3.2.9.1.20 TicksPercentSize

9 Determines the size of lines that divide a Scale. According to the value specified in this property, the default divider line of a Scale appears smaller or bigger. The default value of this property is 20.

4.3.2.10 Dynamic Move

This section contains information about the **properties** of the Dynamic Move object. This object does not have events nor methods associated to it.

4.3.2.10.1 Properties

This section contains information about the properties of the Dynamic Move object.

4.3.2.10.1.1 Detents

The **Detents** property determines the number of steps for this object's movement. Example:

```
Sub CommandButton1_Click()
  MsgBox Screen.Item("DynamicRotate1").Detents
End Sub
```

4.3.2.10.1.2 EnableOverrideLineColor

This property enables or disables this object to override the original color of its image's line by the color defined in the **OverrideLineColor** property. If the **EnableOverrideLineColor** property is enabled, this allows users to modify the original color of this object's line by the color of **OverrideLineColor**. Otherwise, the Picture object displays its original color. Default value of this property is False.

4.3.2.10.1.3 EnableSlider

The **EnableSlider** property enables a slider in object's movement. If this property is enabled, a movement slider is enabled. Otherwise, this does not occur.

4.3.2.10.1.4 OverrideFillColor

When the **OverrideFillMode** property is set to 2 (two) or 3 (three), the **OverrideFillColor** property is used to define a color to use when filling moving objects, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub DrawGroup1_Click()
  ' When clicking this object sets the
  ' Override mode to solid and changes the
  ' fill color to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

4.3.2.10.1.5 OverrideFillMode

The **OverrideFillMode** property specifies a fill mode for moving objects. It changes the original fill mode of an image without changing objects' original fill settings. Possible values for this property are described on the next table.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Object's original fill.

OPTION	DESCRIPTION
1 - WireFrame	Objects are not filled, they only draw their borders.
2 - SolidFill	Objects inside a group are filled with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	Objects inside a group are filled with the color specified in the **OverrideFillColor** property, but it considers the intensity of the original fill color for each object.

Example:

```
Sub DrawGroup1_Click()
  ' When clicking this object sets the
  ' Override mode to solid and changes the
  ' fill color of this image to blue
  OverrideFillMode = 2
  OverrideFillColor = RGB(0, 0, 255)
End Sub
```

4.3.2.10.1.6 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define a color to use on lines of moving objects, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**). Example:

```
Sub Group1_Click()
  OverrideLineColor = RGB(255, 0, 0)
End Sub
```

4.3.2.10.1.7 RangeMax

The **RangeMax** property determines the maximum range for an object's linear sliding.

4.3.2.10.1.8 RangeMin

The **RangeMin** property determines the minimum range for an object's linear sliding.

4.3.2.10.1.9 Value

Indicates an initial value for movement. This must be a value between **RangeMax** and **RangeMin** properties.

4.3.2.11 Dynamic Rotate

This section contains information about the **properties** of the Dynamic Rotate object. This object doe not have events nor methods associated to it.

4.3.2.11.1 Properties

This section contains information about the properties of the Dynamic Rotate object.

4.3.2.11.1.1 Detents

The **Detents** property determines the number of steps for this object's movement.

4.3.2.11.1.2 EnableOverrideLineColor

This property enables or disables this object to override the original color of its image's line by the color defined in the **OverrideLineColor** property. If the **EnableOverrideLineColor** property is enabled, this allows users to modify the original color of this object's line by the color of **OverrideLineColor**. Otherwise, the Picture object displays its original color. Default value of this property is False.

4.3.2.11.1.3 EnableSlider

The **EnableSlider** property enables a slider in object's movement.

4.3.2.11.1.4 OverrideFillColor

When the **OverrideFillMode** property is set to 2 (two) or 3 (three), the **OverrideFillColor** property is used to define a color to use when filling moving objects, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.11.1.5 OverrideFillMode

The **OverrideFillMode** property specifies a fill mode for moving objects. It changes the original fill mode of an image without changing objects' original fill settings. Possible values for this property are described on the next table.

Available options for OverrideFillMode

OPTION	DESCRIPTION
0 - NoOverride	Object's original fill.
1 - WireFrame	Objects are not filled, they only draw their borders.

OPTION	DESCRIPTION
2 - SolidFill	Objects inside a group are filled with the color specified in the **OverrideFillColor** property.
3 - ByBrightness	Objects inside a group are filled with the color specified in the **OverrideFillColor** property, but it considers the intensity of the original fill color for each object

4.3.2.11.1.6 OverrideLineColor

When the **EnableOverrideLineColor** property is set to True, the **OverrideLineColor** property is used to define a color to use on lines of moving objects, instead of their original colors. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property is red (**RGB(255, 0, 0)**).

4.3.2.11.1.7 RangeMax

The **RangeMax** property determines the maximum range for an object's rotational movement.

4.3.2.11.1.8 RangeMin

The **RangeMin** property determines the minimum range for an object's rotational movement.

4.3.2.11.1.9 RotationAngle

This property determines a rotational angle for an object's movement.

4.3.2.11.1.10 RotationDirection

The **RotationDirection** property determines the direction of a rotational angle for an object's movement. Possible values for this property are described on the next table.

Available options for RotationDirection

OPTION	DESCRIPTION
0 - Clockwise	Sets this object's rotational angle to the right
1 - CounterClockWise	Sets this object's rotational angle to the left

4.3.2.11.1.11 Value

Indicates the initial value of a movement. This must be a value between the **RangeMax** and **RangeMin** properties.

4.3.2.12 Microsoft Forms

This section contains information about common **events** and **properties** of Microsoft Forms objects. These objects do not have common methods associated to them.

4.3.2.12.1 Common Events

This section contains information about common events of Microsoft Forms objects.

4.3.2.12.1.1 BeforeDragOver

BeforeDragOver(*Index*, *Cancel*, *Data*, *X*, *Y*, *DragState*, *Effect*, *Shift*)

This event occurs when there is a drag-and-drop action on an object. Use this event to monitor if the mouse pointer entered, left, or stood over a target object. This event is triggered when users move the mouse pointer or press and release any mouse button. Mouse pointer's position indicates the object generating this event. Users can specify the mouse pointer status by checking the *DragState* variable.

Many objects do not support drag-and-drop operations while the *Cancel* variable is set to False, which is the default behavior. This means that an object rejects any attempt of dragging or dropping some other object over it and, therefore, it does not trigger the **BeforeDropOrPaste** event. Text Box and Combo objects are exceptions. These objects accept drag-and-drop operations even when *Cancel* is set to False.

Available parameters on the BeforeDragOver event

PARAMETER	DESCRIPTION
Index	Indicates a page index, in a multi-page object, that is affected by the operation that generated this event. For other objects it is ignored.
Cancel	Event status. Default is False and indicates that the target object handles this event, and not the main application.
Data	Data being dragged to the target object.
X, Y	Position of the mouse pointer inside the target object, in points. *X* is measured from object's left side and *Y* is measured from object's top.

PARAMETER	DESCRIPTION
DragState	Indicates the mouse pointer's condition when this event is generated: • **0 - fmDragStateEnter**: The mouse pointer is inside this object's range • **1 - fmDragStateLeave**: The mouse pointer is outside this object's range • **2 - fmDragStateOver**: The mouse pointer is at a new position, but still inside this object's range
Effect	Indicates all actions that the target object supports, that is, the effect of dragging over this object: • **0 - fmDropEffectNone**: The target object does not accept copying or moving from any origin • **1 - fmDropEffectCopy**: The target object allows copying from any origin to it • **2 - fmDropEffectMove**: The target object allows moving from any origin to it • **3 - fmDropEffectCopyOrMove**: The target object allows copying or moving from any origin to it
Shift	An integer whose sum of factors indicates the status of SHIFT, CTRL, and ALT keys: • **1**: SHIFT key pressed • **2**: CTRL key pressed • **4**: ALT key pressed For example, a value of 5 indicates that SHIFT and ALT keys were pressed (1 + 4 = 5).

4.3.2.12.1.2 BeforeDropOrPaste

BeforeDropOrPaste(*Index*, *Cancel*, *Ctrl*, *Action*, *Data*, *X*, *Y*, *Effect*, *Shift*)

Triggered immediately before a drag-and-drop operation. Normally, it occurs right after a **BeforeDragOver** event.

Available parameters on the BeforeDropOrPaste event

NAME	DESCRIPTION
Index	Indicates a page index, in a multi-page object, that is affected by the operation that generated this event. For other objects it is ignored.
Cancel	Event status. By default it is False and indicates that the target object handles this event, and not the main application.
Ctrl	Target object.

NAME	DESCRIPTION
Data	Data being dragged to the target object.
Action	Indicates the result, based on keyboard settings, of a pending drag-and-drop operation: • **2 - fmActionPaste**: Pastes the selected object on the target object • **3 - fmActionDragDrop**: Indicates that users dragged the selected object from its origin and dropped it on the target object
X, Y	Position of the mouse pointer inside the target object, in points. *X* is measured from object's left side and *Y* is measured from object's top.
DragState	Indicates the mouse pointer's condition when this event is generated: • **0 - fmDragStateEnter**: The mouse pointer is inside this object's range • **1 - fmDragStateLeave**: The mouse pointer is outside this object's range • **2 - fmDragStateOver**: The mouse pointer is at a new position, but still inside this object's range
Effect	Indicates all actions that the target object supports, that is, the effect of dragging over this object: • **0 - fmDropEffectNone**: The target object does not accept copying or moving from any origin • **1 - fmDropEffectCopy**: The target object allows copying from any origin to it • **2 - fmDropEffectMove**: The target object allows moving from any origin to it • **3 - fmDropEffectCopyOrMove**: The target object allows copying or moving from any origin to it
Shift	An integer whose sum of factors indicates the status of SHIFT, CTRL, and ALT keys: • **1**: SHIFT key pressed • **2**: CTRL key pressed • **4**: ALT key pressed For example, a value of 5 indicates that SHIFT and ALT keys were pressed (1 + 4 = 5).

4.3.2.12.1.3 Change

Change()

This event occurs when the value of object's **Value** property changes. Next there are

some examples of actions that trigger the **Change** event:

- Clicking a Check Box, an Option Button, or a Spin Button
- Clicking or selecting words on a List or a Text Editor
- Selecting different tabs on a dialog box
- Moving the scroll bar on a Scrollbar
- Clicking the arrows on a Spin Button
- Selecting different pages on a multi-page object

4.3.2.12.1.4 KeyPress

KeyPress(*KeyAscii*)

This event occurs when an object has keyboard focus and users press a key corresponding to a character that can be displayed on screen (an ANSI key, with its code indicated in the *KeyAscii* variable). That is, this event occurs when any of the following keys are pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- When pressing the TAB key
- When pressing the ENTER key
- When pressing the DEL key (this is not an ANSI key)
- When pressing keyboard arrow keys
- When a key changes focus from one object to another

While a user is pressing a key that produces an ANSI code, an object repeatedly receives the **KeyDown** and **KeyPress** events. When users release that key, the **KeyUp** event occurs. To monitor the physical status of a keyboard or handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

NOTE: For a list with all key codes available for the *KeyAscii* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.3.2.12.1.5 OnError

OnError(*Number*, *Description*, *SCode*, *Source*, *HelpFile*, *HelpContext*, *CancelDisplay*)

Generated by an object's internal error. If this event is not handled, E3 displays a generic error message.

Available parameters on the OnError event

NAME	DESCRIPTION
Number	Integer identifying this error
Description	**String** with this error's description
SCode	Integer with an error code from the OLE subsystem (not used)
Source	**String** with the object that generated this error
HelpFile	**String** with name and path of a help file
HelpContext	Context number of a help topic referring to this error (Integer)
CancelDisplay	**Boolean** indicating whether this error must be displayed on a message box

4.3.2.12.2 Common Properties

This section contains information about common properties of Microsoft Forms objects.

NOTE: E3 uses the Himetric system to define coordinates and widths. In this system, each logical unit is the equivalent of a thousandth of a centimeter, that is, every 1,000 units correspond to one centimeter. Thus, this is the standard adopted in the description of E3 properties, when applicable.

4.3.2.12.2.1 BackColor

Determines a color for an object's background. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property for Combo, List, and Text objects is white (**RGB(255, 255, 255)**), and for all other objects is beige (**RGB(236, 233, 216)**).

4.3.2.12.2.2 ForeColor

Specifies a fill color for object's foreground. In scripts, use VBScript's **RGB** method to create a color to link to this property. Default value of this property for all MS Forms objects is black (**RGB(0, 0, 0)**).

4.3.2.12.2.3 MouseIcon

The **MouseIcon** property sets an image to the mouse pointer, whenever it moves over an object. This property is valid only when the **MousePointer** property is defined as **99 - fmMousePointerCustom**.

An image file can be selected to use as a mouse pointer in two different ways: using the Properties List (.cur or .ico extensions) or via scripts, by using the **LoadPicture** method to specify a file path and name containing a customized icon (.cur extension only). Example:

```
Sub CommandButton1_Click()
  ' Setting the 99 - fmMousePointerCustom item to this property
  ' so that it accepts mouse icon customization
  Screen.Item("CheckBox1").MousePointer = 99
  Screen.Item("CheckBox1").MouseIcon = LoadPicture("C:\a.cur")
End Sub
```

4.3.2.12.2.4 MousePointer

The **MousePointer** property specifies a type of mouse pointer displayed when users move it over an object. Available options for this property are described on the next table.

Available options for MousePointer property

OPTION	DESCRIPTION
0 - fmMousePointerDefault	Default mouse pointer (this image is determined by the object)
1 - fmMousePointerArrow	Arrow-shaped mouse pointer
2 - fmMousePointerCross	Cross-shaped mouse pointer
3 - fmMousePointerBeam	I-beam-shaped mouse pointer
6 - fmMousePointerSizeNesw	Double arrow pointing to North-East and South-West
7 - fmMousePointerSizeNS	Double arrow pointing to North and South
8 - fmMousePointerNWse	Double arrow pointing to North-West and South-East
9 - fmMousePointerWE	Double arrow pointing to West and East
10 - fmMousePointerUpArrow	Arrow pointing up
11 - MousePointerStarHourGlassring	Hourglass
12 - fmMousePointerHelpNoDrop	A **Not** symbol (a circle with a diagonal line) on the upper side of the dragged object (indicates and invalid drop destination)
13 - fmMousePointerAppStarting	Arrow with an hourglass
14 - fmMousePointerHelp	Arrow with a question mark
15 - fmMousePointerSizeAll	Resizes the whole mouse pointer (arrows pointing to North, South, East, and West)

OPTION	DESCRIPTION
99 - fmMousePointerCustom	Uses an icon specified by the **MouseIcon** property

Use the **MousePointer** property to indicate a change in functionality as the mouse pointer slides over Screen objects. For example, an hourglass-shaped pointer (option **11**) is useful to indicate that users must wait for a process or operation to finish. Some icons may vary, depending on system configuration, such as icons linked to desktop themes. Default value of this property is **0 - fmMousePointerDefault**.

4.3.2.12.3 Checkbox

This section contains information about **properties** of the Checkbox object. This object does not have events nor methods associated to it.

4.3.2.12.3.1 Properties

This section contains information about the properties of the Checkbox.

Accelerator

A Defines or retrieves an object's accelerator key. This accelerator key is a shortcut key that, when used together with the ALT key, moves the focus to an object. Default value of this property is empty.

Alignment

The **Alignment** property specifies an object's position relative to its legend. Available options for this property are the following:

- **0 - fmAlignmentLeft**: Aligns the legend to object's left side
- **1 - fmAligmentRight**: Aligns the legend to object's right side

AutoSize

The **AutoSize** property adjusts text width if the available area is larger than object's size. When this property is set to True, text is resized to match object's current size.

BackStyle

The **BackStyle** property defines a style for an object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines an object as transparent, that is, no background is drawn
- **1 - fmBackStyleOpaque**: Defines an object as opaque, that is, its background

is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

Caption

A Defines a text to display on an object.

Font

A The **Font** property is used to determine a font for a Checkbox object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** property of Text, Display, and SetPoint objects.

GroupName

A The **GroupName** property is used to create a mutually exclusive group of objects.

NOTE: This property is not used in E3 and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

The **Locked** property enables or disables object's edition. If this property is set as True, edition is not allowed. Otherwise, users can edit this object. Values configured in the **Enabled** property influence **Locked** behavior. For more details, please check the **Enabled** property. Default value of this property is False.

Picture

The **Picture** property specifies an image (bitmap) set to an object. An image file can be selected in two ways: via Properties List or via scripts, by using the **LoadPicture** method to specify a file path and name containing an image. To remove this image, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not remove this image. Example:

```
Sub CommandButton1_Click()
  Screen.Item("CheckBox1").Picture = LoadPicture("C:\tab.gif")
End Sub
```

PicturePosition

The **PicturePosition** property specifies a picture's position set to an object relative to its legend. Available options for this property are described on the next table.

Available options for the PicturePosition property

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on legend's left side. This legend is aligned to picture's upper side.
1 - fmPicturePositionLeftCenter	Picture appears on legend's left side. This legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on legend's left side. This legend is aligned to picture's lower side.
3 - fmPicturePositionRightTop	Picture appears on legend's right side. This legend is aligned to picture's upper side.
4 - fmPicturePositionRightCenter	Picture appears on legend's right side. This legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on legend's right side. This legend is aligned to picture's lower side.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. This legend is aligned to picture's left side.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. This legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. This legend is aligned to picture's right side.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. This legend is aligned to picture's left side.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. This legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. This legend is aligned to picture's right side.

SpecialEffect

The **SpecialEffect** property specifies an object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmButtonEffectFlat	Object appears flat and has a raised border, a color change, or both.
2 - fmButtonlEffectSunken	Object has a shadow on its upper left side and is raised on its lower right side, as if it is sunken on a Screen.

TextAlign

Specifies how text is aligned on an object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side
- **2 - fmTextAlignCenter**: Centers the text to object's right side
- **3 - fmTextAlignRight**: Aligns the text to object's right side

TripleState

The **TripleState** property determines up to three value statuses for an object. If this property is set to True, users can select between three status options: False, True, or Null. The Null value is displayed as a shaded button. Otherwise, users can only select between True and False values. Default value of this property is False.

Value

Indicates an object's initial value and its behavior is **Boolean**: If True, this object starts as selected, otherwise it starts deselected. Default value of this property is False.

WordWrap

Enables or disables a line break in a text, if the available area for that text exceeds an object's boundaries.

4.3.2.12.4 Option Button

This section contains information about **properties** of the Option Button object. This object does not have events nor methods associated to it.

4.3.2.12.4.1 Properties

This section contains information about the properties of the Option Button.

Accelerator

Defines or retrieves an object's accelerator key. This accelerator key is a shortcut key that, when used together with the ALT key, moves the focus to an object. Default value of this property is empty.

Alignment

The **Alignment** property specifies an object's position relative to its legend. Available options for this property are the following:

- **0 - fmAlignmentLeft**: Aligns the legend to object's left side
- **1 - fmAligmentRight**: Aligns the legend to object's right side

AutoSize

The **AutoSize** property adjusts text width if the available area is larger than object's size. When this property is set to True, text is resized to match object's current size.

BackStyle

The **BackStyle** property defines a style for an object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines an object as transparent, that is, no background is drawn
- **1 - fmBackStyleOpaque**: Defines an object as opaque, that is, its background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

Caption

Defines a text to display on an object.

Font

The **Font** property is used to determine a font for an Option Button object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

GroupName

The **GroupName** property is used to create a mutually exclusive group of objects.

NOTE: This property is not used in E3 and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

The **Locked** property enables or disables object's edition. If this property is set to True, edition is not allowed. Otherwise, users can edit this object. Values configured in the **Enabled** property influence **Locked** behavior. For more details, please check the **Enabled** property. Default value of this property is False.

Picture

The **Picture** property specifies an image (bitmap) set to an object. An image file can be selected in two ways: via Properties List or via scripts, by using the **LoadPicture** method to specify a file path and name containing an image. To remove this image, click the value of the **Picture** property and press the DEL key. The

BACKSPACE key does not remove this image. Example:

```
Sub CommandButton1_Click()
  Screen.Item("OptionButton1").Picture = LoadPicture("C:
\tab.gif")
End Sub
```

PicturePosition

The **PicturePosition** property specifies a picture's position set to an object relative to its legend. Available options for this property are described on the next table.

Available options for the PicturePosition property

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on legend's left side. This legend is aligned to picture's upper side.
1 - fmPicturePositionLeftCenter	Picture appears on legend's left side. This legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on legend's left side. This legend is aligned to picture's lower side.
3 - fmPicturePositionRightTop	Picture appears on legend's right side. This legend is aligned to picture's upper side.
4 - fmPicturePositionRightCenter	Picture appears on legend's right side. This legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on legend's right side. This legend is aligned to picture's lower side.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. This legend is aligned to picture's left side.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. This legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. This legend is aligned to picture's right side.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. This legend is aligned to picture's left side.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. This legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. This legend is aligned to picture's right side.

SpecialEffect

The **SpecialEffect** property specifies an object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmButtonEffectFlat	Object appears flat and has a raised border, a color change, or both.
2 - fmButtonlEffectSunken	Object has a shadow on its upper left side and is raised on its lower right side, as if it is sunken on a Screen.

TextAlign

Specifies how text is aligned on an object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side
- **2 - fmTextAlignCenter**: Centers the text to object's right side
- **3 - fmTextAlignRight**: Aligns the text to object's right side

TripleState

The **TripleState** property determines up to three value statuses for an object. If this property is set to True, users can select between three status options: False, True, or Null. The Null value is displayed as a shaded button. Otherwise, users can only select between True and False values. Default value of this property is False.

Value

Indicates an object's initial value. Its behavior is **Boolean**: If True, this object starts as selected, otherwise it starts deselected. Default value of this property is False.

WordWrap

Enables or disables a line break in the text, if the available area for the text exceeds the boundaries determined by the object.

4.3.2.12.5 Combo

This section contains information about **events**, **methods**, and **properties** of the Combo object.

4.3.2.12.5.1 Events

This section contains information about the events of the Combo object.

DropButtonClick

DropButtonClick()

This event occurs when a Combo is displayed or hidden after clicking an object.

4.3.2.12.5.2 Methods

This section contains information about the methods of the Combo object.

AddItem

AddItem([*pvargItem*[, *pvargIndex*]])

The **AddItem** method is used to add items to a Combo. *pvargItem* is a **String** containing a text to add to a list. If omitted, an empty **String** is added. *pvargIndex* is the index of a text on a list. If omitted, *pvargItem* is added as the last item on this list. Example:

```
Sub CommandButton1_Click()
  EntryCount = EntryCount + 1
  ComboBox1.AddItem(EntryCount & " - Selection")
End Sub
```

Clear

Clear()

Clears an object's text.

Copy

Copy()

Copies the previously selected text to the Clipboard. Use the **Paste** method to paste this text into the indicated place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Copy()
End Sub
```

Cut

Cut()

Cuts the previously selected text to the Clipboard. Use the **Paste** method to paste this text into the indicated place. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Cut()
End Sub
```

DropDown

DropDown()

The **DropDown** method opens a list of items from a Combo. Calling this method has the same effect as clicking, at run time, the object's right side button. Example:

```
Sub CommandButton1_Click()
  Dim ComboBox1
  ComboBox1.AddItem "Pineapple"
  ComboBox1.AddItem "Strawberry"
```

```
  ComboBox1.AddItem "Grape"
  ComboBox1.AddItem "Orange"
  ComboBox1.DropDown()
End Sub
```

Paste

Paste()

Inserts Clipboard's content into an object. Example:

```
Sub CommandButton1_Click()
  Screen.Item("ComboBox1").Paste()
End Sub
```

RemoveItem

RemoveItem(*pvargIndex*)

Removes items from a Combo. This method has the *pvargIndex* parameter, which specifies a row to remove, starting at 0 (zero). That is, the first element is 0 (zero), the second element is 1 (one), and so on. Example:

```
Sub CommandButton2_Click()
  ComboBox1.SetFocus
  ' Check if the list contains selected data
  If ComboBox1.ListCount >= 1 Then
    ' If there is no selection,
    ' selects the last item.
    If ComboBox1.ListIndex = -1 Then
      ComboBox1.ListIndex = ComboBox1.ListCount - 1
    End If
    ComboBox1.RemoveItem(ComboBox1.ListIndex)
  End If
End Sub
```

4.3.2.12.5.3 Properties

This section contains information about the properties of the Combo object.

AutoSize

◩ The **AutoSize** property adjusts text width if the available area is larger than object's size. When this property is set to True, text is resized to match object's current size.

AutoTab

◩ The **AutoTab** property enables or disables automatic tab on an object. If this property is set to True, automatic tab occurs. Otherwise, it is not used.

After users type the maximum number of characters in an object (by using the **MaxLength** property), focus moves automatically to the next object in the tab order, when reaching these characters. For example, for a Combo to always display stock data with five characters, users can use the **MaxLength** property to specify the

maximum number of characters to type in this object and the **AutoTab** property to automatically move to the next object after users type five characters.

AutoWordSelect

Enables or disables automatic selection of words on an object. If this property is set to True, the indicated word is selected in the text plus the next space, if users selected part of it. Otherwise, only the character indicated in the word is selected.

BackStyle

The **BackStyle** property defines a style for this object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines this object as transparent, that is, no background is drawn
- **1 - fmBackStyleOpaque**: Defines this object as opaque, that is, its background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

BorderColor

This property determines a border color applied to an object. With this property, users can apply a default color or customize it by editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value of this property is black (**RGB(0, 0, 0)**).

BorderStyle

The **BorderStyle** property determines a border style applied to an object. Available options are the following:

- **0 - fmBorderStyleNone**: No border
- **1 - fmBorderStyleSingle**: Single border

BoundColumn

Determines a column on this Combo where data is stored. For example, if each row contains eight items and the **BoundColumn** property is set to 3 (three), this object stores information on the third column of the currently selected row. If its value is set to 0 (zero), this value is passed to object's **ListIndex** property. If its value is equal to 1 (one) or higher, indicated data is attributed to the column referring to the value specified in this property. Columns start at 1 (one).

NOTE: This property has no effect on E3 and it is kept for compatibility reasons with Microsoft Forms standard specification.

CanPaste

The **CanPaste** property specifies whether the Clipboard contains data supported by this object. If this property is set to True, an object can receive information pasted from the Clipboard. If data from Clipboard is in a format that an object does not support, the value of the **CanPaste** property is set to False. For example, when trying to paste a bitmap to an object that only supports text, **CanPaste** is False. This property is only available at run time.

Column

Specifies an object's row and column. If only a column's value is specified, the **Column** property reads or writes the column specified on object's current row. For example, "MyComboBox.Column(3)" reads or writes object's third column. This property is only available at run time.

ColumnCount

The **ColumnCount** property specifies the number of columns on an object. Configuring **ColumnCount** as 0 (zero) does not display any column, and configuring it as -1 (minus one) displays all available columns. Default value of this property is 1 (one).

ColumnHeads

The **ColumnHeads** property enables or disables the display of column titles on an object. If this property is set to True, a title is displayed. Otherwise, column titles are not displayed. Default value is False.

ColumnWidths

The **ColumnWidths** property is used to specify an object's column width, in points. A value equal to -1 (minus one) or blank forces the width to be calculated on the column (the minimum width on a calculated column is 72 points, or one inch). A value equal to 0 (zero) hides a column. To produce narrower columns, users must specify a width in this property or use one of the values described on the next table.

Available options for the ColumnWidths property

OPTION	DESCRIPTION
90;72;90	The first column is 90 points (1.25 inches), the second column is 72 points (one inch), and the third column is 90 points.
6 cm;0;6 cm	The first column is 6 centimeters, the second column is hidden, and the third column is 6 centimeters. As part of the third column is visible, a horizontal scroll bar is visible.

OPTION	DESCRIPTION
1.5 in;0;2.5 in	The first column is 1.5 inches, the second column is hidden, and the third column is 2.5 inches.
2 in;;2 in	The first column is 2 inches, the second column is 1 inch (default), and the third column is 2 inches. As only half of the third column is visible, a horizontal scroll bar is visible.
(Empty)	All three columns have the same width (1.33 inches). Default value of this property is empty (E3 uses system's default value).

CurTargetX

9 Returns the horizontal insertion position of a text in an object. This position is measured in Himetric units (one himeter is equal to 0.0001 of a meter). Users can use either **CurTargetX** or **CurX** properties to move the insertion point in a text as users move over an object's content. When users move the insertion point to another row in a text, the **CurTargetX** property specifies the best position for text's insertion point. The **CurX** property is defined in this value if text row is greater than the value of **CurTargetX**. Otherwise, the **CurX** property is defined at the end of text's row. This property is only available at run time.

NOTE: This property has no effect on E3 and it is kept for compatibility reasons with the Microsoft Forms standard specification.

CurX

9 The **CurX** property specifies the current horizontal position of an object's insertion point. This property is applied to an object with several rows, that is, whose **Multiline** property is enabled. Returned value is valid when this object has focus. Users can use the **Multiline** property and the **CurX** property to place a text's insertion point as users use the scroll bar on object's content. When users move the insertion point to another row in a text by scrolling object's content, the **CurTargetX** property specifies the most appropriate position for text's insertion point. The **CurX** property is defined with this value if the text row is greater than the value of **CurTargetX**. Otherwise, **CurX** property is defined at the end of text's row. This property is only available at run time.

DragBehavior

Enables or disables a feature of dragging and dropping a text in object's content. Available options for this property are the following:

- **0 - fmDragBehaviorDisabled**: Does not allow dragging and dropping a text in object's content
- **1 - fmDragBehaviorEnabled**: Allows dragging and dropping a text in object's

content

Default value of this property is **0 - fmDragBehaviorDisabled**.

NOTE: The **DragBehavior** property has no effect if the **Style** property is set to **2 - fmStyleDropDownList**.

DropButtonStyle

This property specifies a symbol to display on Combo's button. Available options for this property are described on the next table.

Available options for the DropButtonStyle property

OPTION	DESCRIPTION
0 - fmDropButtonStylePlain	Displays a flat button, without a symbol
1 - fmDropButtonStyleArrow	Displays a down arrow (default)
2 - fmDropButtonStyleEllipsis	Displays an ellipsis
3 - fmDropButtonStyleReduce	Displays a horizontal line with an underscore character

Default value of this property is **1 - fmDropButtonStyleArrow**.

EnterFieldBehavior

This property controls how text's content is selected in the edition area when the TAB key is pressed on an object, and not when an object receives focus as the result of a **SetFocus** method. Available options for this property are the following:

- **0 - fmEnterFieldBehaviorSelectAll**: Selects the whole text's content when the TAB key is pressed on an object
- **1 - fmEnterFieldBehaviorRecallSelection**: Selection remains unchanged

Default value of this property is **0 - fmEnterFieldBehaviorSelectAll**.

Font

The **Font** property is used to determine a font for a Combo object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** property of Text, Display, and SetPoint objects.

HideSelection

The **HideSelection** property specifies whether the selected text is still highlighted when an object loses its focus. If this property is set to True, the selected text only remains highlighted if an object has the focus. Otherwise, the selected text always appears highlighted, regardless of object's focus. Default value of this property is True.

IMEMode

The **IMEMode** property specifies an object's IME (*Input Method Editor*) mode.

NOTE: This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese) and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification.

Available options for this property are described on the next table.

Available options for the IMEMode property

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default)
1 - fmIMEModeOn	IME active
2 - fmIMEModeOff	IME deactivated. English mode
3 - fmIMEModeDisable	IME mode deactivated. Users cannot activate IME mode via keyboard
4 - fmIMEModeHiragana	IME active with full width Hiragana mode
5 - fmIMEModeKatakanaFull	IME active with full width Katakana mode
6 - fmIMEModeKatakana	IME active with half width Katakana mode
7 - fmIMEModeAlphaFull	IME active with full width Alphanumeric mode
8 - fmIMEModeAlpha	IME active with half width Alphanumeric mode
9 - fmIMEModeHangulFull	IME active with full width Hangul mode
10 - fmIMEModeHangul	IME active with half width Hangul mode
11 - fmIMEModeHanziFull	IME active with full width Hanzi mode
12 - fmIMEModeHanzi	IME active with half width Hanzi mode

LineCount

The **LineCount** property returns the number of rows of an object. This property is only available at run time.

List

Returns or defines column and row entries on an object's list. Column and row numbering starts at 0 (zero), that is, the row number of the first row on a list is 0 (zero) and the column number of the first column is 0 (zero). The number of the second row or column is 1 (one), and so on. This property is only available at run time.

ListCount

Returns the number of items on an object's list. This property is only available at run time.

ListIndex

9 Identifies the item currently selected on a list, which is called an index. **ListIndex** values range from -1 (minus one) to the total amount of rows on a list minus one (that is, **ListCount** - 1). When no row is selected, **ListIndex** returns -1 (minus one). When users select a row on a Combo, the system defines the value of **ListIndex** property. The value of **ListIndex** property on the first row of a list is 0 (zero), the value of the second row is 1 (one), and so on. This property is only available at run time.

ListRows

9 The **ListRows** property determines the maximum amount of rows on an object's list. Default value of this property is 8 (eight).

ListStyle

The **ListStyle** property determines an object's list style. Available options for this property are the following:

- **0 - fmListStylePlain**: List with background items highlighted
- **1 - fmListStyleOption**: Displays option buttons or combo boxes for a list with several options. When users select one item on a group, the linked option button is selected and the option buttons for the other group items are deselected

Default value of this property is **0 - fmListStylePlain**.

ListWidth

9 The **ListWidth** property determines the width of an object's list. Default value of this property is 0 (zero).

Locked

The **Locked** property enables or disables object edition. If this property is set to True, edition is not allowed. Otherwise, users can edit an object. Values configured in the **Enabled** property influence **Locked** behavior. For more details, please check the **Enabled** property. Default value of this property is False.

MatchEntry

Searches, using a user-typed text, for a text entry that matches data in an object. When a text match is found, its row is selected and its column's content is displayed. Available options are the following:

- **0 - fmMatchEntryFirstLetter**: Searches for a text entry that matches the first character typed in an object. If the same character is repeatedly typed, it moves to the next text entry that starts with that character, and so on
- **1 - fmMatchEntryComplete**: As every character is typed, this object searches

for a text entry that matches the typed character

- **2 - fmMatchEntryNone**: Does not perform a search in this object

Default value of this property is **1 - fmMatchEntryComplete**.

MatchFound

◩ Indicates whether a user-typed text matches any entry on a list. If this property is set to True, the content of the **Value** property matches one of the records on a list. Otherwise, the content of the **Value** property does not match any of the records on a list (default).

This property is available only at run time, and does not apply when the **MatchEntry** property is defined as 2 (two). Default value of this property is False.

MatchRequired

◩ Specifies whether a user-typed text must match existing items on a Combo. If this property is set to True, users cannot leave a Combo until the text inserted matches an item on this object. Otherwise, the inserted text on a Combo can be different from all existing data on its list.

MaxLength

9 The **MaxLength** property determines the maximum number of characters on an object. Configuring this property as 0 (zero), there is no limit of characters on this object.

SelectionMargin

◩ Enables or disables a selection margin on this object. If this property is set to True, text is selected when clicking object's margin. Otherwise, text is not selected when clicking object's margin.

NOTE: If the **SelectionMargin** property is set to True when an object is printed,its selection margin is also printed.

SelLength

9 Returns the number of selected characters on an object. This property is only available at run time.

SelStart

9 Indicates a starting point for the selected text or an insertion point if no text is selected. This property is only available at run time.

SelText

A Returns the selected text on an object. This property is only available at run time.

ShowDropButtonWhen

The **ShowDropButtonWhen** property specifies when to display a drop button (object's browsing key). Available options for this property are the following:

- **0 - fmShowDropButtonWhenNever**: Does not display a drop button
- **1 - fmShowDropButtonWhenFocus**: Only displays a drop button when this object has focus
- **2 - fmShowDropButtonWhenAlways**: Always displays a drop button

SpecialEffect

The **SpecialEffect** property specifies object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object appears flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on its upper left side and a shadow on its lower right side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on its upper left side and raised on its lower right side. This object and its border appear sunken on a Screen.
3 - fmSpecialEffectEtched	Border looks etched around object edges.
6 - fmSpecialEffectBump	Object has a ledge on its lower right side and looks flat on its upper left side.

Style

Determines an object's style. Available options are the following:

- **0 - fmStyleDropDownCombo**: A Combo behaves as a drop down combo. Users may type a value on its edition area or select a value from its drop down list (default)
- **2 - fmStyleDropDownList**: A Combo behaves as a list and users must select a value from its list

Text

Returns the text of the selected option. This property is only available at run time.

TextAlign

Specifies how text is aligned on this object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side

- **2 - fmTextAlignCenter**: Aligns the text to object's center
- **3 - fmTextAlignRight**: Aligns the text to object's right side

TextColumn

The **TextColumn** property identifies a column on an object. Values for the **TextColumn** property range from -1 (minus one) to the number of columns on this list. The **TextColumn** value for the first column is 1 (one), its value for the second column is 2 (two), and so on. Configuring the **TextColumn** property as 0 (zero) displays values for the **ListIndex** property. Configuring the **TextColumn** property as -1 (minus one) displays the first column that has the value of the **ColumnWidths** property greater than 0 (zero).

TextLength

Returns the number of characters typed on an object. This property is only available at run time.

TopIndex

The **TopIndex** property defines or returns a list item that appears on top of this list. This property returns -1 (minus one) if this list is empty or it is not displayed.

Value

This is the value of the **BoundColumn** property of the currently selected rows. A change in the value of the **Value** property does not imply on changing the value of the **BoundColumn** property. To add or delete entries on a combo, users can use either **AddItem** or **RemoveItem** methods.

4.3.2.12.6 Command Button

This section contains information about **events** and **properties** of the Command Button object. This object does not have methods associated to it.

4.3.2.12.6.1 Events

This section contains information about the events of the Command Button object.

MouseMove

MouseMove()

This event occurs when the mouse pointer is moved over a Command Button.

4.3.2.12.6.2 Properties

This section contains information about the properties of the Command Button object.

Accelerator

A Defines or gets object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, moves the focus to an object. Default value of this property is an empty **String**.

AutoSize

◩ The **AutoSize** property adjusts text width, if the available area exceeds object's size. For a Command Button object, when this property is set to True, text width is resized to fit the current object's size. Text content is cut when it exceeds object's area.

BackStyle

The **BackStyle** property defines a style for object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines an object as transparent, that is, no object's background is drawn
- **1 - fmBackStyleOpaque**: Defines an object as opaque, that is, its background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

Caption

A Defines a text to display on an object.

Font

A The **Font** property is used to determine a font for a Command Button object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

Locked

◩ The **Locked** property enables or disables object's edition. If this property is set to True, edition is not allowed. Otherwise, users can edit an object. Values configured in the **Enabled** property influence **Locked** behavior. For more details, please check the **Enabled** property. Default value of this property is False.

Picture

The **Picture** property specifies an image (bitmap) attributed to an object. An image file can be selected in two ways: via Properties List or via scripts, by using the **LoadPicture** function to specify a file path and name containing an image. To remove an image, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not remove an image.

PicturePosition

The **PicturePosition** property specifies a position for a picture set to an object relative to its legend. Available options for this property are described on the next table.

Available options for the PicturePosition property

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on legend's left side. This legend is aligned to picture's upper side.
1 - fmPicturePositionLeftCenter	Picture appears on legend's left side. This legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on legend's left side. This legend is aligned to picture's lower side.
3 - fmPicturePositionRightTop	Picture appears on legend's right side. This legend is aligned to picture's upper side.
4 - fmPicturePositionRightCenter	Picture appears on legend's right side. This legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on legend's right side. This legend is aligned to picture's lower side.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. This legend is aligned to picture's left side.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. This legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. This legend is aligned to picture's right side.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. This legend is aligned to picture's left side.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. This legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. This legend is aligned to picture's right side.

TakeFocusOnClick

Specifies whether an object receives focus when clicked. If this property is set to True, an object receives focus when clicked. Otherwise, an object does not receives focus.

WordWrap

Enables or disables a line break on a text, if the available area for a text exceeds the boundaries determined by an object.

4.3.2.12.7 Label

This section contains information about **properties** of the Label object. This object does not have events nor methods associated to it.

4.3.2.12.7.1 Properties

This section contains information about the properties of the Label object.

Accelerator

A Defines or retrieves an object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, moves the focus to an object. Default value of this property is an empty **String**.

AutoSize

The **AutoSize** property adjusts text width, if the available area exceeds object's size. For a Label object, when this property is set to True, text is resized to match object's current size, thus allowing its full display.

BackStyle

The **BackStyle** property defines a style for an object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines this object as transparent, that is, no object background is drawn
- **1 - fmBackStyleOpaque**: Defines this object as opaque, that is, its background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

BorderColor

This property determines a color to apply to an object's border. With this property, users can apply the default color or customize it while editing. For this property to be applicable, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value of this property is black (**RGB(0, 0, 0)**).

BorderStyle

The **BorderStyle** property defines a border style to apply to an object. Available options are:

- **0 - fmBorderStyleNone**: No border
- **1 - fmBorderStyleSingle**: Single border

Caption

A Defines a text to display on an object.

Font

A The **Font** property is used to determine a font for a Label object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

Picture

The **Picture** property specifies a picture (bitmap) attributed to an object. A picture file can be selected in two ways: via Properties List or via scripts, by using the **LoadPicture** function to specify a file path and name containing a picture. To remove a picture, click the value of the **Picture** property and press the DEL key. The BACKSPACE key does not remove a picture.

PicturePosition

The **PicturePosition** property specifies a picture position attributed to an object relative to its legend. Available options for this property are described on the next table.

Available options for PicturePosition

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on legend's left side. This legend is aligned to picture's upper side.
1 - fmPicturePositionLeftCenter	Picture appears on legend's left side. This legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on legend's left side. This legend is aligned to picture's lower side.
3 - fmPicturePositionRightTop	Picture appears on legend's right side. This legend is aligned to picture's upper side.
4 - fmPicturePositionRightCenter	Picture appears on legend's right side. This legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on legend's right side. This legend is aligned to picture's lower side.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. This legend is aligned to picture's left side.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. This legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. This legend is aligned to picture's right side.

OPTION	DESCRIPTION
9 - fmPicturePositionBelowLeft	Picture appears below the legend. This legend is aligned to picture's left side.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. This legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. This legend is aligned to picture's right side.

SpecialEffect

The **SpecialEffect** property specifies an object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object appears flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on its upper left side and a shadow on its lower right side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on its upper left side and raised on its lower right side. Object and its border appear sunken on a Screen.
3 - fmSpecialEffectEtched	Border looks etched around object edges.
6 - fmSpecialEffectBump	Object has a ledge on its lower right side and looks flat on its upper left side.

TextAlign

Specifies how text is aligned on an object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side
- **2 - fmTextAlignCenter**: Aligns the text to object's center
- **3 - fmTextAlignRight**: Aligns the text to object's right side

WordWrap

Enables or disables a line break on a text, if the available area for this text exceeds the boundaries determined by an object. For this property to work, the **Multiline** property must be set to True.

4.3.2.12.8 List

This section contains information about **methods** and **properties** of the List object. This object does not have events associated to it.

4.3.2.12.8.1 Methods

This section contains information about the methods of the List object.

AddItem

AddItem([*pvargItem*[, *pvargIndex*]])

The **AddItem** method is used to add items to a List. *pvargItem* is a **String** containing a text to add to a List and, if omitted, adds an empty **String**. *pvargIndex* is the index of a text on a List and, if omitted, *pvargItem* is added as the last item on a List. Example:

```
Sub CommandButton1_Click()
  EntryCount = EntryCount + 1
  ListBox1.AddItem(EntryCount & " - Selection")
End Sub
```

Clear

Clear()

Clears an object's text.

RemoveItem

RemoveItem(*pvargIndex*)

Removes items from a List. This method has the *pvargIndex* parameter, which specifies an item to exclude, starting at 0 (zero). That is, the first element is 0 (zero), the second element is 1 (one), and so on. Example:

```
Sub CommandButton2_Click()
  List1.SetFocus
  ' Checks if this list has selected items
  If List1.ListCount >= 1 Then
    ' If there is no selection,
    ' selects the last item on this list.
    If List1.ListIndex = -1 Then
      List1.ListIndex = List1.ListCount - 1
    End If
    List1.RemoveItem(List1.ListIndex)
  End If
End Sub
```

4.3.2.12.8.2 Properties

This section contains information about the properties of the List object.

BorderColor

This property defines a color for a border applied to an object. With this property, users can apply a default color or customize it while editing. For this property to be applicable, the **BorderStyle** property must be set to **1 -**

fmBorderStyleSingle. Default value of this property is black (**RGB(0, 0, 0)**).

BorderStyle

The **BorderStyle** property determines a border style applied to an object. Available options are:

- **0 - fmBorderStyleNone**: No border
- **1 - fmBorderStyleSingle**: Single border

BoundColumn

Determines a column on a list where data is stored. For example, if each row contains eight items and the **BoundColumn** property is set to 3 (three), application stores information on the third column of the currently selected row. If value is set to 0 (zero), this value is then passed to object's **ListIndex** property. If value is set to 1 (one) or higher, indicated data is attributed to the column referring to the value specified in this property. Columns start at 1 (one).

NOTE: This property has no effect on E3 and it is kept for compatibility reasons with Microsoft Forms standard specification.

Column

Specifies an object's row and column. If only a column value is specified, the **Column** property reads or writes a column specified on object's current row. For example, "MyListBox.Column(3)" reads or writes object's third column. This property is only available at run time.

ColumnCount

The **ColumnCount** property specifies the number of columns of an object. Configuring **ColumnCount** as 0 (zero) does not display any column, and configuring this property as -1 (minus one) displays all available columns. Default value of this property is 1 (one).

ColumnHeads

The **ColumnHeads** property enables or disables displaying column titles on an object. If this property is set to True, a title is displayed. Otherwise, column titles are not displayed. Default value is False.

ColumnWidths

The **ColumnWidths** property is used to specify object's column width, in points. A value equal to -1 (minus one) or blank forces a width to be calculated on the column (the minimum width on a calculated column is 72 points, or one inch). A value equal to 0 (zero) hides a column. To produce narrower columns, users must specify a width in this property or use one of the values described on the next table.

Available options for the ColumnWidths property

OPTION	DESCRIPTION
90;72;90	The first column is 90 points (1.25 inches), the second column is 72 points (one inch), and the third column is 90 points.
6 cm;0;6 cm	The first column is 6 centimeters, the second column is hidden, and the third column is 6 centimeters. As part of the third column is visible, a horizontal scroll bar is visible.
1.5 in;0;2.5 in	The first column is 1.5 inches, the second column is hidden, and the third column is 2.5 inches.
2 in;;2 in	The first column is 2 inches, the second column is 1 inch (default), and the third column is 2 inches. As only half of the third column is visible, a horizontal scroll bar is visible.
(Empty)	All three columns have the same width (1.33 inches). Default value of this property is empty (E3 uses system's default value).

Font

A The **Font** property is used to determine a font for a List object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

IMEMode

The **IMEMode** property specifies the object's IME (*Input Method Editor*) mode.

NOTE: This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese) and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification.

The available options are described on the next table.

Available options for the IMEMode property

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default)
1 - fmIMEModeOn	IME active
2 - fmIMEModeOff	IME deactivated. English mode
3 - fmIMEModeDisable	IME deactivated. Users cannot activate IME mode via keyboard

OPTION	DESCRIPTION
4 - fmIMEModeHiragana	IME active with full width Hiragana mode
5 - fmIMEModeKatakanaFull	IME active with full width Katakana mode
6 - fmIMEModeKatakana	IME active with half width Katakana mode
7 - fmIMEModeAlphaFull	IME active with full width Alphanumeric mode
8 - fmIMEModeAlpha	IME active with half width Alphanumeric mode
9 - fmIMEModeHangulFull	IME active with full width Hangul mode
10 - fmIMEModeHangul	IME active with half width Hangul mode
11 - fmIMEModeHanziFull	IME active with full width Hanzi mode
12 - fmIMEModeHanzi	IME active with half width Hanzi mode

IntegralHeight

The **IntegralHeight** property adjusts the height of text's editing area, if the available area exceeds the object's size. If this property is set to True, the height of text's editing area is adjusted to match the object's current size, thus allowing a full display of text's content. Otherwise, text's editing area keeps its original size. If a text is larger than the available space, it is not displayed in the object.

List

Returns or defines column and row entries on the object's list. Column and row numbering starts at 0 (zero). That is, the row number of the first row on the list is 0 (zero) and the column number of the first column is 0 (zero). The number of the second row or column is 1 (one), and so on. This property is only available at run time.

ListCount

Returns the number of items in the object's list. This property is only available at run time.

ListIndex

Identifies the item currently selected on the list, which is called index. The values of **ListIndex** range from -1 (minus one) to the total amount of rows on a list minus one (that is, **ListCount** - 1). When no row is selected, **ListIndex** returns -1 (minus one). When users select a row on a List, the application defines the value of **ListIndex** property. The value of **ListIndex** property for the first row on a list is 0 (zero), the value of the second row is 1 (one), and so on. This property is only available at run time.

ListStyle

The **ListStyle** property determines the object's list style. The available options for this property are the following:

- **0 - fmListStylePlain**: List with background items highlighted

- **1 - fmListStyleOption**: Displays option buttons or combo boxes for a list with several options. When users select one item on the group, the option button associated to this item is selected and the option buttons for the other group items are deselected

The default value of this property is **0 - fmListStylePlain**.

NOTE: The **1 - fmListStyleOption** option can only be enabled if the **MultiSelect** property is set to **1 - fmMultiselectMulti**.

Locked

The **Locked** property enables or disables object edition. If this property is set to True, edition is not allowed. Otherwise, users can edit the object. Values configured for the **Enabled** property influence the behavior of **Locked**. For more details, please check the **Enabled** property. Default value of this property is False

MatchEntry

Searches, using a user-typed text, for a text entry that matches data in the object. When a text instance is found, the row with that text is highlighted and the column's content is displayed. The available options are the following:

- **0 - fmMatchEntryFirstLetter**: Searches for a text entry that matches the first character typed in the object. If the same character is repeatedly typed, it moves to the next text entry that starts with that character, and so on
- **1 - fmMatchEntryComplete**: As every character is typed, the object searches for a text entry that matches the typed character
- **2 - fmMatchEntryNone**: Does not perform a search in the object

The default value of this property is **1 - fmMatchEntryComplete**.

MultiSelect

The **MultiSelect** property indicates whether the object allows multiple selections. The available options for this property are the following:

- **0 - fmMultiSelectSingle**: Only one item can be selected
- **1 - fmMultiSelectMulti**: Allows selecting one item by pressing the space bar or with a mouse click, thus selecting or deselecting an item on the list
- **2 - fmMultiSelectExtended**: Allows selecting one item by pressing the SHIFT key, with a mouse click, or by pressing the SHIFT key and one of the arrow keys, extending the selection to the current item. Pressing the CTRL key and clicking the mouse, selects or deselects an item

The default value of this property is **0 - fmMultiSelectSingle**.

Selected

Selects or deselects an item, and checks whether an item is selected, when the **Multiline** property is set to True. To check whether an item is selected, the item's index to check must be provided and then the property returns whether it is selected or not. So, it is possible to determine which items are selected when users select more than one item. This property is only available at run time. When users are not using multiple selections, it is recommended to use the **Value** or **ListIndex** properties.

SpecialEffect

The **SpecialEffect** property specifies the object's appearance. The available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object appears flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on its upper left side and a shadow on its lower right side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on its upper left side and raised on its lower right side. This object and its border appear sunken on a Screen.
3 - fmSpecialEffectEtched	Border looks etched around object edges.
6 - fmSpecialEffectBump	Object has a ledge on its lower right side and looks flat on its upper left side.

Text

A Returns the text of the selected option. This property is only available at run time.

TextAlign

Specifies how text is aligned on the object. The available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to the object's left side
- **2 - fmTextAlignCenter**: Aligns the text to the object's center
- **3 - fmTextAlignRight**: Aligns the text to the object's right side

TextColumn

The **TextColumn** property identifies a column in the object. Values for this property range from -1 (minus one) to the number of columns on the list. The value of the **TextColumn** property for the first column is 1 (one), for the second column is

2 (two), and so on. Configuring the **TextColumn** property as 0 (zero) displays values for the **ListIndex** property. Configuring the **TextColumn** property as -1 (minus one) displays the first column that has the value of the **ColumnWidths** greater than 0 (zero).

TopIndex

The **TopIndex** property defines or returns the list item that appears on top of the list. This property returns -1 (minus one) if the list is empty or is not displayed.

Value

This is the value of the **BoundColumn** property of the currently selected rows. This property has no effect in E3, and is kept for compatibility reasons with Microsoft Forms standard specification.

4.3.2.12.9 Toggle Button

This section contains information about **events** and **properties** of the Toggle Button. This object does not have methods associated to it.

4.3.2.12.9.1 Events

This section contains information about the events of the Toggle Button object.

MouseMove

MouseMove()

This event occurs when the mouse pointer is moved over a Toggle Button object.

4.3.2.12.9.2 Properties

This section contains information about the properties of the Toggle Button object.

Accelerator

Defines or retrieves an object's accelerator key. This accelerator key is a shortcut key that, when used with the ALT key, moves the focus to an object. Default value of this property is an empty **String**.

Alignment

The **Alignment** property specifies an object's position, relative to its legend. Available options for this property are the following:

- **0 - fmAlignmentLeft**: Aligns the legend to object's left side
- **1 - fmAligmentRight**: aligns the legend to object's right side

AutoSize

The **AutoSize** property adjusts text width, if the available area is larger than the object's size. For a Toggle Button object, when this property is set to True, text is resized to match the object's current size, thus allowing its full display.

BackStyle

The **BackStyle** property defines a style for object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines an object as transparent, that is, no object background is drawn
- **1 - fmBackStyleOpaque**: Defines an object as opaque, that is, background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

Caption

Defines a text to display on this object.

Font

The **Font** property is used to determine a font for a Toggle Button object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

GroupName

The **GroupName** property is used to create a mutually exclusive group of objects. This property is only available at run time.

NOTE: This property is not used on E3 and it is kept for compatibility reasons with Microsoft Forms standard specification.

Locked

The **Locked** property enables or disables object edition. If this property is set to True, edition is not allowed. Otherwise, users can edit this object. Values configured for the **Enabled** property influence **Locked**. behavior. For more details, please check the **Enabled** property. Default value of this property is False.

Picture

The **Picture** property specifies a picture (bitmap) set to an object. A picture file can be selected in two ways: via Properties List or via scripts, by using the **LoadPicture** function to specify a file path and name containing this picture. To

remove a picture, click the value of **Picture** property and press the DEL key. The BACKSPACE key does not delete a picture.

PicturePosition

The **PicturePosition** property specifies the position of a picture set to an object relative to its legend. Available options for this property are described on the next table.

Available options for the PicturePosition property

OPTION	DESCRIPTION
0 - fmPicturePositionLeftTop	Picture appears on legend's left side. This legend is aligned to picture's upper side.
1 - fmPicturePositionLeftCenter	Picture appears on legend's left side. This legend is centered relative to the picture.
2 - fmPicturePositionLeftBottom	Picture appears on legend's left side. This legend is aligned to picture's lower side.
3 - fmPicturePositionRightTop	Picture appears on legend's right side. This legend is aligned to picture's upper side.
4 - fmPicturePositionRightCenter	Picture appears on legend's right side. This legend is centered relative to the picture.
5 - fmPicturePositionRightBottom	Picture appears on legend's right side. This legend is aligned to picture's lower side.
6 - fmPicturePositionAboveLeft	Picture appears above the legend. This legend is aligned to picture's left side.
7 - fmPicturePositionAboveCenter	Picture appears above the legend. This legend is centered below the picture (default).
8 - fmPicturePositionAboveRight	Picture appears above the legend. This legend is aligned to picture's right side.
9 - fmPicturePositionBelowLeft	Picture appears below the legend. This legend is aligned to picture's left side.
10 - fmPicturePositionBelowCenter	Picture appears below the legend. This legend is centered above the picture.
11 - fmPicturePositionBelowRight	Picture appears below the legend. This legend is aligned to picture's right side.

SpecialEffect

The **SpecialEffect** property specifies an object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object appears flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on its upper left side and a shadow on its lower right side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on its upper left side and raised on its lower right side. This object and its border appear sunken on a Screen.
3 - fmSpecialEffectEtched	Border looks etched around object edges.
6 - fmSpecialEffectBump	Object has a ledge on its lower right side and looks flat on its upper left side.

TextAlign

Specifies how text is aligned on an object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side
- **2 - fmTextAlignCenter**: Aligns the text to object's center
- **3 - fmTextAlignRight**: Aligns the text to object's right side

TripleState

The **TripleState** property determines up to three value statuses for an object. If this property is set to True, users can select from three status options: False, True, or Null. The Null value is displayed as a shaded button. Otherwise, users can only select between True and False values. Default value of this property is False.

Value

Indicates this object's initial value. It has a **Boolean** behavior: If True, this object starts as selected, otherwise it starts deselected. Default value of this property is False.

WordWrap

Enables or disables a line break on a text, if the available area for that text exceeds the boundaries determined for an object.

4.3.2.12.10 Text

This section contains information about **events**, **methods**, and **properties** of the Text object.

4.3.2.12.10.1 Events

This section contains information about the events of the Text object.

DropButtonClick

DropButtonClick()

This event occurs when a list of options is displayed or hidden when clicking this object.

4.3.2.12.10.2 Methods

This section contains information about the methods of the Text object.

Copy

Copy()

Copies to the Clipboard a previously selected text. Use the **Paste** method to paste that text in another place.

Cut

Cut()

Cuts to the Clipboard a previously selected text. Use the **Paste** method to paste that text in another place.

Paste

Paste()

Inserts on a text object the Clipboard content.

4.3.2.12.10.3 Properties

This section contains information about the properties of the Text object.

AutoSize

The **AutoSize** property adjusts text width, if the available area exceeds object's size. For a Text object, when this property is set to True, text width is resized to the same object's width. Default value of this property is False.

NOTE: It is advisable to avoid using the **AutoSize** property with a Text that also uses the **Multiline** and **WordWrap** properties. When users type into a Text with these properties set to True, it automatically resizes itself as a long and narrow text box, with a one-character width and a one-text-row size.

AutoTab

The **AutoTab** property enables or disables automatic tab on an object. If this property is set to True, automatic tab occurs. Otherwise, it is not used.

After users type the maximum number of characters on an object (by using the **MaxLength** property), focus moves automatically to the next object in its tab order, whenever these characters are reached. For example, for a Text to always display stock data with five characters, users can use the **MaxLength** property to specify the maximum number of characters to type in an object and the **AutoTab** property to automatically move to the next object after users type five characters.

AutoWordSelect

Enables or disables automatic selection of words on an object. If this property is set to True, the indicated word is selected on a text plus the next space, if users selected part of it. Otherwise, only the character indicated in the word is selected.

BackStyle

The **BackStyle** property defines a style for an object's background. Available options for this property are the following:

- **0 - fmBackStyleTransparent**: Defines this object as transparent, that is, no background is drawn
- **1 - fmBackStyleOpaque**: Defines this object as opaque, that is, background is drawn (default value)

NOTE: This property does no affect bitmap transparency. Users must use an image editor such as Paintbrush, for example, to save a transparent bitmap. Not all ActiveX objects support transparent bitmaps.

BorderColor

This property determines a color for an object's border. With this property, users can apply a default color, or customize it by editing. To apply this property, the **BorderStyle** property must be set to **1 - fmBorderStyleSingle**. Default value of this property is black (**RGB(0, 0, 0)**).

BorderStyle

The **BorderStyle** property determines a style for an object's border. Available options are:

- **0 - fmBorderStyleNone**: No border
- **1 - fmBorderStyleSingle**: Single border

CanPaste

The **CanPaste** property specifies whether the Clipboard contains data supported

by this object. If this property is set to True, this object can receive information pasted from the Clipboard. If data from Clipboard is in a format that this object does not support, the value of the **CanPaste** property is set to False. For example, when trying to paste a bitmap to an object that only supports text, **CanPaste** is False. This property is only available at run time.

CurLine

9 Specifies this object's current row, that is, a row that contains a text's insertion point. The number of the first row is 0 (zero). Default value of this property is 0 (zero).

CurTargetX

9 Returns the horizontal insertion position of a text on an object. This position is measured in Himetric units (a himeter corresponds to 0,0001 meters). Users can use the **CurTargetX** and **CurX** properties to move a text's insertion point while moving through an object's content. When moving this insertion point to another text row, the **CurTargetX** property specifies the best position for the desired text's insertion point. The **CurX** property is defined with this value, if the text row is greater than the value of **CurTargetX**. Otherwise, the **CurX** property is defined as the end of the text row. This property is only accessible at run time.

CurX

9 The **CurX** property specifies the current horizontal position of an object's insertion point. This property is applied to an object with several rows, that is, whenever the **Multiline** property is enabled. The return value is valid when this object receives focus. Users can use either the **Multiline** or the **CurX** properties to place a text's insertion point, as users use the scroll bar through object's content. When users move the insertion point to another row on a text by scrolling object's content, the **CurTargetX** property specifies the best position for this text's insertion point. The **CurX** property is defined with this value if text row is greater than the value of **CurTargetX**. Otherwise, the **CurX** property is defined at the end of the text row. This property is only available at run time.

DragBehavior

◩ Enables or disables a feature of dragging and dropping a text on an object's content. Available options for this property are the following:

- **0 - fmDragBehaviorDisabled**: Does not allow dragging and dropping a text on an object's content
- **1 - fmDragBehaviorEnabled**: Allows dragging and dropping a text on an object's content

Default value of this property is **0 - fmDragBehaviorDisabled**.

NOTE: The **DragBehavior** property has no effect if the **Style** property is set to **2 - fmStyleDropDownList**.

EnterFieldBehavior

This property controls how text content is selected on an edition area when the TAB key is pressed on an object, and not when this object receives focus as the result of a **SetFocus** method. Available options for this property are the following:

- **0 - fmEnterFieldBehaviorSelectAll**: Selects the whole content of a text when the TAB key is pressed on an object (default)
- **1 - fmEnterFieldBehaviorRecallSelection**: Selection remains unchanged

EnterKeyBehavior

Defines the behavior of the ENTER key on an object. If this property is set to True, when users press the ENTER key that creates a new line on an object's text edition area. Otherwise, when users press the ENTER key focus goes to the next object in its tab order. This also occurs if the **Multiline** property is set to False, regardless of the value of **EnterKeyBehavior** property.

The combination of CTRL + ENTER keys also depends on the value of the **Multiline** property. If this property is set to True, when those two keys are pressed that creates a new line on an object's text edition area, regardless of the value of **EnterKeyBehavior** property. If this property is set to False, these keys have no effect on a text.

Font

The **Font** property is used to determine a font for a Text object. This property cannot be used in Links and contains the same sub-properties described in the **TextFont** of Text, Display, and SetPoint objects.

HideSelection

The **HideSelection** property specifies whether the selected text is still highlighted when an object does not have focus anymore. If this property is set to True, the selected text is not highlighted, unless this object has focus. Otherwise, the selected text always appears highlighted. Default value of this property is True.

IMEMode

The **IMEMode** property specifies an object's IME (*Input Method Editor*) mode. This property only applies to applications written in Eastern languages (Chinese Simplified and Traditional, Korean, and Japanese), and it is ignored in other applications. It was kept for compatibility reasons with Microsoft Forms standard specification. Available options are described on the next table.

Available options for the IMEMode property

OPTION	DESCRIPTION
0 - fmIMEModeNoControl	No IME control (default)

OPTION	DESCRIPTION
1 - fmIMEModeOn	IME mode active
2 - fmIMEModeOff	IME mode deactivated. English mode
3 - fmIMEModeDisable	IME mode deactivated. Users cannot activate IME mode via keyboard
4 - fmIMEModeHiragana	IME mode active with full width Hiragana mode
5 - fmIMEModeKatakanaFull	IME mode active with full width Katakana mode
6 - fmIMEModeKatakana	IME mode active with half width Katakana mode
7 - fmIMEModeAlphaFull	IME mode active with full width Alphanumeric mode
8 - fmIMEModeAlpha	IME mode active with half width Alphanumeric mode
9 - fmIMEModeHangulFull	IME mode active with full width Hangul mode
10 - fmIMEModeHangul	IME mode active with half width Hangul mode
11 - fmIMEModeHanziFull	IME mode active with full width Hanzi mode
12 - fmIMEModeHanzi	IME mode active with half width Hanzi mode

IntegralHeight

◩ The **IntegralHeight** property adjusts the height of a text's editing area, if the available area exceeds this object's size. If this property is set to True, the height of text's editing area is adjusted to match this object's current size, thus allowing a full display of text's content. Otherwise, text's editing area keeps its original size. If a text is larger than the available space, it is not displayed on an object.

LineCount

9 The **LineCount** property returns the number of lines on this object. This property is only available at run time.

Locked

◩ The **Locked** property enables or disables object's edition. If this property is set to True, edition is not allowed. Otherwise, users are allowed to edit this object. Values configured for the **Enabled** property influence **Locked** behavior. For more details, please check the **Enabled** property. Default value of this property is False.

MaxLength

9 The **MaxLength** property determines the maximum number of characters on an object. By configuring this property as 0 (zero), there is no limit of characters on an object.

Multiline

The **Multiline** property indicates whether a text has multiple lines (True) or it is a simple text box (False). This can be viewed when Viewer is running. Default value of this property is False.

PasswordChar

Converts an object's text to a special character, configured in this property. Use this property to protect sensitive information, such as passwords or security codes. The **PasswordChar** property's value is a character (usually an asterisk) that appears on an object, instead of user-typed real characters. If a character is not specified, this control displays user-typed characters.

ScrollBars

Specifies whether this object has vertical or horizontal scroll bars, or even both. Available options are the following:

- **0 - fmScrollBarNone**: No scroll bars are displayed
- **1 - fmScrollBarHorizontal**: A horizontal scroll bar is displayed
- **2 - fmScrollBarVertical**: A vertical scroll bar is displayed

Default value of this property is **0 - fmScrollBarNone**.

SelectionMargin

Enables or disables an object's selection margin. If this property is set to True, text is selected when users click an object's margin. Otherwise, text is not selected when users click an object's margin.

NOTE: If the **SelectionMargin** property is set to True when printing this object, selection margin is also printed.

SelLength

Returns the number of selected characters on an object. This property is only available at run time.

SelStart

Indicates a starting point of the selected text or the insertion point, if no text is selected. This property is only available at run time.

SelText

Returns the text selected on an object. This property is only available at run time.

SpecialEffect

The **SpecialEffect** property specifies an object's appearance. Available options for this property are described on the next table.

Available options for the SpecialEffect property

OPTION	DESCRIPTION
0 - fmSpecialEffectFlat	Object appears flat and has a raised border, a color change, or both.
1 - fmSpecialEffectRaised	Object is raised on its upper left side and a shadow on its lower right side, as a relief.
2 - fmSpecialEffectSunken	Object has a shadow on its upper left side and raised on its lower right side. This object and its border appear sunken on a Screen.
3 - fmSpecialEffectEtched	Border looks etched around object edges.
6 - fmSpecialEffectBump	Object has a ledge on its lower right side and looks flat on its upper left side.

TabKeyBehavior

Determines whether tabs are allowed on edition area. If this property is set to True, pressing the TAB key inserts a space character on edition area. Otherwise, pressing the TAB key sets focus to the next object in object's tab order.

Text

Returns a text being typed in the selected option. This property is only available at run time.

TextAlign

Specifies how text is aligned on an object. Available options are the following:

- **1 - fmTextAlignLeft**: Aligns the text to object's left side
- **2 - fmTextAlignCenter**: Aligns the text to object's center
- **3 - fmTextAlignRight**: Aligns the text to object's right side

TextLength

Returns the number of characters typed on an object. This property is only available at run time.

Value

This is the text on object's edition area. This is a **Variant**-type property, and it may assume values of any type (**Date**, **Boolean**, **String**, etc.).

WordWrap

Enables or disables a line break on a text, if the available area for this text exceeds the boundaries determined by an object. For this property to work, the

Multiline property must be set to True.

4.3.2.12.11 Spin Button

This section contains information about **events** and **properties** of the Spin Button object. This object does not have methods associated to it.

4.3.2.12.11.1 Events

This section contains information about the events of the Spin Button object.

SpinDown

SpinDown()

Occurs when users press the down arrow key. This event decreases object's **Value** property.

SpinUp

SpinUp()

Occurs when users press the up arrow key. This event increases object's **Value** property.

4.3.2.12.11.2 Properties

This section contains information about the properties of the Spin Button object.

Delay

9 Specifies a delay time for this object. The **Delay** property affects the time duration between consecutive **SpinUp**, **SpinDown**, and **Change** events, generated when users click a Spin Button and keep it pressed. The first event occurs immediately. The delay time until the second occurrence of this event is five times the value specified for this property. After this initial time, the interval between events is the value specified for **Delay**.

Default value for **Delay** is 50 ms. This means that this object starts the first event after 250 ms (five times the specified value), and starts each subsequent event after 50 ms.

Max

9 The **Max** property is used to determine object's maximum limit.

Min

9 The **Min** property is used to determine object's minimum limit.

Orientation

The **Orientation** property is used to determine object's orientation on Screen. Available options for this property are the following:

- **-1 - fmOrientationAuto**: Determines object's orientation automatically based on its dimensions, that is, how it was created
- **0 - fmOrientationVertical**: This object is placed vertically
- **1 - fmOrientationHorizontal**: This object is placed horizontally

Default value of this property is **-1 - fmOrientationAuto**.

SmallChange

The **SmallChange** property specifies the amount of movement that occurs when users click an object's scroll arrow. Default value of this property is 1 (one).

Value

An integer between the values defined for **Min** and **Max** properties. It indicates the initial position for incrementing or decrementing. It does not accept values lower than **Min** nor higher than **Max**.

4.3.2.12.12 Scrollbar

This section contains information about **events** and **properties** of the Scrollbar object. This object does not have methods associated to it.

4.3.2.12.12.1 Events

This section contains information about the events of the Scrollbar object.

Scroll

Scroll()

Generated when a scroll bar pointer moves to any direction.

4.3.2.12.12.2 Properties

This section contains information about the properties of the Scrollbar object.

Delay

Specifies a delay time for this object. The **Delay** property affects the time duration between consecutive **SpinUp**, **SpinDown**, and **Change** events, generated when users click a Scrollbar and keep it pressed. The first event occurs immediately. The delay time until the second occurrence of this event is five times the value specified for this property. After this initial time, the interval between events is the value specified for **Delay**.

Default value for **Delay** is 50 ms. This means that this object starts the first event after 250 ms (five times the specified value), and starts the subsequent events after 50 ms.

LargeChange

Specifies the amount of steps for Scrollbar's cursor. The **LargeChange** property's value is the amount that the **Value** property changes when users click the area between Scrollbar's box and Scrollbar's cursor. Any integer value is allowed for the **LargeChange** property, but the recommended interval is from -32,767 to +32,767, as this value must be between the values determined by Scrollbar's **Max** and **Min** properties.

Max

The **Max** property determines object's maximum limit.

Min

The **Min** property determines object's minimum limit.

Orientation

The **Orientation** property is used to determine object's orientation on Screen. Available options for this property are the following:

- **-1 - fmOrientationAuto**: Determines object's orientation automatically based on its dimensions, that is, how it was created
- **0 - fmOrientationVertical**: This object is placed vertically
- **1 - fmOrientationHorizontal**: This object is placed horizontally

Default value of this property is **-1 - fmOrientationAuto**.

ProportionalThumb

The **ProportionalThumb** property specifies whether a Scrollbar size is equal to its dimension. If this property is set to True, Scrollbar's box has the same dimensions of this object. Otherwise, if this object is resized, Scrollbar's box remains with its original size. Default value of this property is True.

SmallChange

The **SmallChange** property specifies the amount of movement that occurs when users click an object's scroll arrow. Default value of this property is 1 (one).

Value

An integer between values defined for **Min** and **Max** properties. It indicates a Scrollbar's initial position. It does not accept values less than **Min** nor greater than **Max**.

4.3.2.13 E2Controls

This section contains information about events, methods, and properties of E2Control objects.

4.3.2.13.1 Common Properties

This section contains information about the common properties of the E2Control objects.

4.3.2.13.1.1 Frame_BorderColor

Defines a color for this object's frame.

4.3.2.13.1.2 Frame_BorderEnabled

Enables or disables this object's frame.

4.3.2.13.1.3 Frame_BorderThickness

Defines a thickness for this object's frame, in pixels.

4.3.2.13.1.4 Frame_Color

Defines a background color for this object's title area. Default value of this property is gray (**RGB(192, 192, 192)**).

4.3.2.13.1.5 Frame_Enable

Enables or disables displaying this object's frame. Default value of this property is True, except for an **E2Button** object.

4.3.2.13.1.6 Frame_Enable3D

Enables or disables a 3D effect for this object's frame.

4.3.2.13.1.7 Frame_Separator

Enables or disables displaying a separator line between the title and the object.

4.3.2.13.1.8 Frame_Set3DInset

If this property is set to True, this object's border appears as lowered. If it is False (default), this object's border appears as raised.

4.3.2.13.1.9 Frame_Thickness3D

9 Defines a thickness for a 3D border of this object's frame, in pixels.

4.3.2.13.1.10 Frame_Title

A This property defines a title for this object's frame.

4.3.2.13.1.11 Frame_TitleColor

Defines a font color for frame's title. Default value of this property is black (**RGB(0, 0, 0)**).

4.3.2.13.1.12 Frame_TitleEnabled

Enables or disables displaying a frame's title. Default value of this property is True.

4.3.2.13.1.13 Frame_TitleFont

A The **Frame_TitleFont** property is used to determine a font for frame's title. This property cannot be used in scripts or Links, being set only via Studio.

4.3.2.13.2 E2Animation

This section contains information about **properties** of the E2Animation object. This object does not have events nor methods associated to it.

4.3.2.13.2.1 Properties

This section contains information about the properties of the E2Animation object.

BackgroundColor

This property defines a background color for an E2Animation object. Default value of this property is white (**RGB(255, 255, 255)**).

BlinkTime

9 Defines a time interval, in milliseconds, for this object's blinking effect.

Border

Enables or disables displaying a border around this object.

DefaultZone

9 Defines a default Zone, which is displayed when the associated Tag is out of the limits of other Zones defined for this object.

IsTransparent

If the value of this property is True, defines that this object's background is transparent, allowing that screen's background appears. Otherwise, this background color is solid, as defined in the **BackgroundColor** property.

Value

This property defines a value that determines the active Zone. Default value of this property is 0 (zero).

Zones

Collection of Zones of an E2Animation object.

4.3.2.13.2.2 Zone Collection

This section contains information about **methods** of the Zone Collection object. This object does not have events nor methods associated to it.

Methods

This section contains information about the methods of the Zone Collection object.

Add

Add([*AxisName*])

Adds a new Zone in the Zone Collection. The *AxisName* parameter is optional and has no effect, being kept for compatibility reasons with earlier versions.

Remove

Remove(*Index*)

Removes a Zone. The *Index* parameter indicates a Zone index to remove.

Zones

Defines a set of bitmap images used to create an animation effect on this object. Zones can be configured by accessing object's Properties window, on its **E2Animation** tab. Available options for this tab are described on the next table.

Available options for E2Animation tab

OPTION	DESCRIPTION
Zones	List with all Zones defined on this object.
Add	Adds a new Zone.
Remove	Deletes the selected Zone.
Default zone	Defines the selected Zone as object's default Zone.

OPTION	DESCRIPTION
Blink	Defines if a bitmap blinks when this object's value is inside this Zone's interval.
Tip	Displays a help text about this Zone.
Minimum	Minimum value for this Zone's variation.
Maximum	Maximum value for this Zone's variation.
Image file	Name of a bitmap file that is displayed when this object's value is inside this Zone's interval.
Example	Displays a preview of a bitmap file of the selected Zone.

Properties

This section contains information about the properties of the Zones of the E2Animation object.

Blink

◩ Indicates that this Zone is part of a blinking effect. Default value of this property is False.

Filename

A Indicates a name of an image file used in this Zone.

Maximum

9 Defines a maximum value for this Zone. Default value of this property is 20000.

Minimum

9 Defines a minimum value for this Zone. Default value of this property is 0 (zero).

TipEnable

◩ Enables or disables a tip for this Zone. Default value of this property is False.

TipText

A Defines a text for this Zone's tip. Default value of this property is an empty **String**.

4.3.2.13.3 E2Bitmap

This section contains information about **properties** of the E2Bitmap object. This object does not have events nor methods associated to it.

4.3.2.13.3.1 Properties

This section contains information about the properties of the E2Bitmap object.

Filename

Defines a name for a picture file linked to this E2Bitmap. This file's path may be a full file path on disk, as well as a relative application path (when this picture file is inserted as an application Resource). Default value of this property is an empty **String**.

IsTransparent

This property enables or disables this object's transparency, based on a color defined by the **TransparentColor** property.

TransparentColor

Defines a color considered by the **IsTransparent** property as transparent. Default value of this property is white (**RBG(255, 255, 255)**).

4.3.2.13.4 E2Button

This section contains information about **events** and **properties** of the E2Button object. This object does not have methods associated to it.

4.3.2.13.4.1 Events

This section contains information about the events of the E2Button object.

OnRelease

OnRelease()

This event is triggered when the mouse button is released.

4.3.2.13.4.2 Properties

This section contains information about the properties of the E2Button object.

Action

This property defines an E2Button's behavior when clicked. Available values for this property are:

- **0 - Momentary**: Standard button behavior, appearing lowered only when pressing this button
- **1 - Toggle**: It has two states, turned on and turned off
- **2 - Jog**: Switches between two values, one when this button is pressed and another one when this button is released

Default value of this property is **0 - Momentary**.

Alignment

Determines a text alignment for an E2Button. Possible values for this property are:

- **0 - HorizontalAlignmentLeft**: Aligns the text to the left
- **1 - HorizontalAlignmentCenter**: Aligns the text to the center
- **2 - HorizontalAlignmentRight**: Aligns the text to the right

Default value of this property is **1 - HorizontalAlignmentCenter**.

BackgroundColor0

Defines a background color for an E2Button when it is not pressed. Default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundColor1

Defines a background color for an E2Button when it is pressed. Default value of this property is gray (**RGB(192, 192, 192)**).

Bitmap0

Defines an E2Button's picture when it is not pressed. Default value of this property is an empty **String**.

Bitmap1

Defines an E2Button's picture when it is pressed. Default value of this property is an empty **String**.

Text0

Defines an E2Button's text when it is not pressed. Default value of this property is "OFF".

Text1

Defines an E2Button's text when it is pressed. Default value of this property is "ON".

TextColor0

Defines a color for an E2Button's text when it is not pressed. Default value of this property is black (**RGB(0, 0, 0)**).

TextColor1

Defines a color for an E2Button's text when it is pressed. Default value of this property is black (**RGB(0, 0, 0)**).

TextFont0

This property is used to determine an E2Button's font when it is not pressed. This property cannot be used in scripts or Links, being set only via Studio.

TextFont1

This property is used to determine an E2Button's font when it is pressed. This property cannot be used in scripts or Links, being set only via Studio.

Type

Defines an E2Button's type. Possible values for this property are the following:

- **0 - ButtonTypeKey**: Standard button behavior
- **1 - ButtonTypeSwitchH**: This button's behavior is a switch horizontally divided
- **2 - ButtonTypeSwitchV**: This button's behavior is a switch vertically divided
- **3 - ButtonTypeLeverH**: This button's behavior is a lever that moves from left to right and vice versa
- **4 - ButtonTypeLeverV**: This button's behavior is a lever that moves from top to bottom and vice versa
- **5 - ButtonTypeTransparent**: This button is transparent
- **6 - ButtonTypeUserBitmap**: This button alternates pictures defined in the **Bitmap0** and **Bitmap1** properties
- **7 - ButtonTypeCheckbox**: This button's behavior is the same as a Check Box
- **8 - ButtonTypeRadio**: This button's behavior is the same as a Radio Box

Default value of this property is **0 - ButtonTypeKey**.

Value

The **Value** property is a **Variant** that assumes the value in the **Value0** property if this button is not pressed, and the value in the **Value1** property if this button is pressed.

Value0

Defines the value of the **Value** property when this button is not pressed.

Value1

Defines the value of the **Value** property when this button is pressed.

4.3.2.13.5 E2Display

This section contains information about **properties** of the E2Display object. This object does not have events nor methods associated to it.

4.3.2.13.5.1 Properties

This section contains information about the properties of the E2Display object.

BackgroundColor

This property defines a color for this object's background. Default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundStyle

Defines a style for this object's background. Available values for this property are the following:

- **0 - bsTransparent**: Background is transparent
- **1 - bsOpaque**: The color defined in the **BackgroundColor** property is visible

Default value of this property is **1 - bsOpaque**.

Format

Contains a text that represents a mask where object values are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies a number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays a number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal,

or binary, for example)

- **Other**: Allows directly editing a format code, or selecting a previously created format

HorizontalAlignment

Defines a horizontal alignment for an E2Display's text. Available values for this property are the following:

- **0 - HorizontalAlignmentLeft**: Aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: Aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: Aligns horizontally to the right

Default value of this property is **1 - HorizontalAlignmentCenter**.

MultiLine

Defines whether an object has multiple lines or not. This property is only available if the **Value** property is a **String**-type.

TextColor

Defines a color for this object's text. Default value of this property is black (**RGB(0, 0, 0)**).

TextFont

The **TextFont** property is used to determine this object's font. This property cannot be used in Links. Please check the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

Value

This property contains a **Variant** that can assume values from any data type, and the way these values are displayed is defined in the **Format** property.

VerticalAlignment

Defines a vertical alignment for an E2Display's text. Available values for this property are the following:

- **0 - VerticalAlignmentTop**: Aligns vertically with object's upper side
- **1 - VerticalAlignmentMiddle**: Aligns vertically with object's center
- **2 - VerticalAlignmentBottom**: Aligns vertically with object's bottom side

Default value of this property is **1 - VerticalAlignmentMiddle**.

4.3.2.13.6 E2Gauge

This section contains information about **properties** of the E2Gauge object. This object does not have events nor methods associated to it.

4.3.2.13.6.1 Properties

This section contains information about the properties of the E2Gauge object.

BackgroundColor

This property defines a color for this object's background. Default value of this property is gray (**RGB(128, 128, 128)**).

BulletsVisible

Displays or hides **bullet**-type marks on the scale.

DecimalPlaces

This property defines the number of decimal places for this E2Gauge's nominal value.

FrameColor

Defines a color for this object's background.

HiColorLegend

Defines a legend color for a **High** limit. Default value of this property is yellow (**RGB(255, 255, 0)**).

HiDiv

Sets the beginning of a scale for a **High** limit. Default value of this property is 13300.

HiHiColorLegend

Defines a legend color for a **Very High** limit. Default value of this property is red (**RGB(255, 0, 0)**).

HiHiDiv

Sets the beginning of a scale for a **Very High** limit. Default value of this property is 16600.

HiHiLimitVisible

Enables or disables displaying a **Very High** limit.

HiLimit

9 The maximum value of this property is 1 (one) and the minimum is limited by the **LowLimit** property. Default value of this property is 0.7.

HiLimitVisible

◩ Enables or disables displaying a **High** limit.

LegendVisible

◩ Displays a bar along an E2Gauge object, which allows configuring different colors, depending on a range of values. Default value of this property is True.

LimitVisible

◩ Defines whether a scale's minimum and maximum values appear on a chart or not.

LowColorLegend

Defines a legend color for a **Low** limit. Default value of this property is dark green (**RGB(0, 128, 0)**).

LowDiv

9 Sets the beginning of a scale for a **Low** limit. Default value of this property is 6600.

LowLimit

9 The minimum value for this property is 0.1 and the maximum is limited by the **HiLimit** property. Default value of this property is 0.62.

LowLimitVisible

◩ Enables or disables displaying a **Low** limit.

LowLowColorLegend

Defines a legend color for a **Very Low** limit. Default value of this property is green (**RGB(0, 255, 0)**).

LowLowDiv

9 Sets the beginning of a scale for a **Very Low** limit. Default value of this property is 3300.

LowLowLimitVisible

◩ Enables or disables displaying a **Very Low** limit.

Maximum

9 Defines the maximum value of an E2Gauge's scale.

Minimum

9 Defines the minimum value of an E2Gauge's scale.

NeedleColor

Defines a color for an E2Gauge's needle. Default value of this property is white (**RGB(255, 255, 255)**).

NeedleThickness

Defines the thickness of an E2Gauge needle, in pixels. Default value of this property is 2 (two), and it only accepts values 1 (one) or 2 (two).

NormalColor

Defines a legend color for a **Normal** limit. Default value of this property is olive (**RGB(128, 128, 0)**).

NumberOfPoints

Defines the number of subdivisions visible on an object's scale.

Orientation

Defines an E2Gauge's orientation. Possible values for this property are the following:

- **0 - Left**: This object's lower side aligns with frame's left side
- **1 - Up**: This object's lower side aligns with frame's upper side
- **2 - Down**: This object's lower side aligns with frame's lower side
- **3 - Right**: This object's lower side aligns with frame's right side

Default value of this property is **2 - Down**.

Reverted

Enables or disables reversing this object's scale.

ShowFrame

Enables or disables displaying a background along the needle's path.

StartAngle

Defines an initial display angle for an E2Gauge's needle.

SubTickColor

Defines a color for a scale's subdivision. Default value of this property is black (**RGB(0, 0, 0)**).

SubTicksVisible

Enables or disables displaying scale's subdivisions.

TextColor

Defines a color for a scale's text. Default value of this property is black (**RGB(0, 0, 0)**).

TextFont

The **TextFont** property is used to determine a font for a scale's text. This property cannot be used in Links. Please check the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

ThickTicks

Enables or disables displaying thicker scale dividers. Default value of this property is False.

TickColor

Defines a color for scale's dividers. Default value of this property is black (**RGB(0, 0, 0)**).

TicksVisible

Enables or disables displaying scale dividers.

TickValues

Enables or disables displaying values for scale dividers.

TotalNumberOfSubTicks

Defines the total amount of subdivisions displayed on a scale.

Value

This property defines a value between the **Maximum** and **Minimum** properties of this object's scale.

ValueVisible

Enables or disables displaying a value from the **Value** property. Default value of this property is False.

4.3.2.13.7 E2Setpoint

This section contains information about **properties** of the E2Setpoint object. This object does not have events nor methods associated to it.

4.3.2.13.7.1 Properties

This section contains information about the properties of the E2Setpoint object.

AutoSend

If this property is True (default), the value defined in the **Value** property is updated on Links, as soon as this object loses focus. Otherwise, Links only receive this value when the ENTER key is pressed.

BackgroundColor

This property defines a color for this object's background. Default value of this property is gray (**RGB(192, 192, 192)**).

BackgroundStyle

Defines a style for this object's background. Values for this property are the following:

- **0 - bsTransparent**: Background is transparent
- **1 - bsOpaque**: The color defined in the BackgroundColor property is visible

Default value of this property is **1 - bsOpaque**.

EnableMaxLimit

Enables or disables the definition of a maximum limit for E2Setpoint's value.

EnableMinLimit

Enables or disables the definition of a minimum limit for E2Setpoint's value.

Format

Contains a text that represents a mask where object values are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers as signed or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies a number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays a number with a mantissa and exponent notation. Ideal

for numbers with variable magnitude. Allows up to 15 decimal places

- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing a format code, or selecting a previously created format

HorizontalAlignment

Defines a horizontal alignment for an E2Setpoint's text. Available values for this property are the following:

- **0 - HorizontalAlignmentLeft**: Aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: Aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: Aligns horizontally to the right

Default value of this property is **1 - HorizontalAlignmentCenter**.

HScroll

Enables or disables displaying a horizontal scroll bar on a text, if the **MultiLine** property is set to True.

MaxLimit

9 Maximum limit that can be reached by this object's **Value** property. Default value of this property is 200. This limit is only checked if the **EnableMaxLimit** property is enabled.

MinLimit

9 Minimum limit that can be reached by this object's **Value** property. Default value of this property is 0 (zero). This limit is only checked if the **EnableMinLimit** property is enabled.

MultiLine

Defines whether this object has multiple lines or not. This property is only available if the **Value** property is of **String**-type.

ReadOnly

Indicates whether this object can be edited or not at run time. Default value of this property is False.

Refresh

Indicates whether this E2Setpoint's value is updated or not, as soon as a Tag value changes. Default value of this property is True.

SelectAllOnFocus

Enables or disables selecting all characters of an E2Setpoint when this object receives focus. Default value of this property is True.

TextColor

Defines a color for this object's text. Default value of this property is black (**RGB(0, 0, 0)**).

TextFont

The **TextFont** property is used to determine an object's font. This property cannot be used in Links. Please check the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

Type

Defines a type for this Setpoint. Available values for this property are the following:

- **0 - setpointString**: Accepts any alphanumerical characters
- **1 - setpointNumeric**: Accepts only numerical characters and a decimal separator (dot or comma, according to regional settings)
- **2 - setpointDateTime**: Accepts only date and time values, which are converted to a format defined on regional settings

Default value of this property is **1 - setpointNumeric**.

Value

This property defines a value for an E2Setpoint. The way to display this value is defined in the **Format** property.

VerticalAlignment

Defines a vertical alignment for an E2Setpoint's text. Available values for this property are the following:

- **0 - VerticalAlignmentTop**: Aligns vertically to object's top
- **1 - VerticalAlignmentMiddle**: Aligns vertically to object's center
- **2 - VerticalAlignmentBottom**: Aligns vertically to object's bottom

Default value of this property is **1 - VerticalAlignmentMiddle**.

VScroll

Enables or disables displaying a vertical scroll bar on a text, if the **MultiLine** property is set to True.

4.3.2.13.8 E2Text

This section contains information about **properties** of the E2Text object. This object does not have events nor methods associated to it.

4.3.2.13.8.1 Properties

This section contains information about the properties of the E2Text object.

BlinkTime

Defines a time interval, in milliseconds, for this object's blinking effect.

DefaultZone

Defines a default Zone for this object.

Value

This property contains a **Variant** that may assume values from any data type (**Integer**, **Boolean**, **String**, etc.).

Zones

Collection of Zones of an E2Text object.

4.3.2.13.8.2 Zone Collection

This section contains information about the **methods** of the Zone Collection object of the E2Text. This object does not have events nor properties associated to it.

Methods

This section contains information about the methods of the Zone Collection object of the E2Text.

Add

Add([*AxisName*])

Adds a new Zone in the Zone Collection. The *AxisName* parameter is optional and has no effect, being kept for compatibility reasons with earlier versions.

Remove

Remove(*Index*)

Removes a Zone. The *Index* parameter indicates a Zone index to remove.

Zones

Defines a set of Zones for an E2Text object. These Zones can be configured by accessing object's Properties window, on its **Zones** tab. Available options for this

tab are described on the next table.

Available options for Zones tab

OPTION	DESCRIPTION
Zones	List with all Zones defined for this object.
Add	Adds a new Zone.
Remove	Removes the selected Zone.
Blinks every (ms)	Defines whether text and background of a Zone blink when this object's value is inside a Zone's interval.
Message	Text of a message displayed when this object's value is inside a Zone's interval.
Alignment	Defines a text alignment.
Font	Defines a text font.
Background Color	Defines a color for Zone's background.
Transparent	Defines if this object's background is transparent when this Zone is active.
Default zone	Defines the selected Zone as object's default Zone.
Blink	Defines if this Zone blinks when this object's value is inside a Zone's interval.
Minimum	Minimum value for this Zone.
Maximum	Maximum value for this Zone.
Tip	Displays a help text for this Zone.
Example	Displays an example of this Zone's behavior at run time.

Properties

This section contains information about the properties of the E2Text object Zones.

BackgroundColor

Defines a background color for this Zone's text. Default value of this property is white (**RGB(255, 255, 255)**).

Blink

Indicates that this Zone participates in the blinking effect. Default value of this property is False. If it is enabled, this Zone switches with the default Zone, according to a time defined in E2Text's **BlinkTime** property.

HorizontalAlignment

Defines a text alignment. Available values are:

- **0 - HorizontalAlignmentLeft**: Aligns horizontally to the left
- **1 - HorizontalAlignmentCenter**: Aligns horizontally to the center
- **2 - HorizontalAlignmentRight**: Aligns horizontally to the right

Default value of this property is **1 - HorizontalAlignmentCenter**.

Maximum

9 Defines a maximum value for this Zone. Default value of this property is 20000.

Message

A Defines a text linked to this Zone. This message is displayed when the associated Tag is inside the limits of this Zone.

Minimum

9 Defines a minimum value for this Zone. Default value of this property is 0 (zero).

TextColor

Defines a color for this Zone's text. Default value of this property is black (**RGB(0, 0, 0)**).

TextFont

A Defines a style, a color, and a size for a font used to display this message's text. This property cannot be used in Links. Please check the **TextFont** property of Text, Display, and SetPoint objects for more information about sub-properties that can be modified via scripts.

TipEnable

Enables or disables a tip for this Zone. Default value of this property is False.

TipText

A Defines a text for this Zone's tip. Default value of this property is an empty **String**.

Transparent

Defines that this object's background is transparent when this Zone is active.

4.3.2.14 Elipse KeyPad

This section contains information about **methods** and **properties** of the Elipse KeyPad object. This object does not have events associated to it.

4.3.2.14.1 Methods

This section contains information about the methods of the Elipse KeyPad object.

4.3.2.14.1.1 Hide

Hide()

Hides Elipse KeyPad. This method has no effect if KeyPad is already invisible.

4.3.2.14.1.2 Show

Show()

Displays Elipse KeyPad. This method has no effect if KeyPad is already visible.

4.3.2.14.2 Properties

This section contains information about the properties of the Elipse KeyPad object.

4.3.2.14.2.1 AutoHideOnEnter

◩ Automatically hides KeyPad when the virtual keyboard's ENTER key is pressed.

4.3.2.14.2.2 AutoHideOnEsc

◩ Automatically hides KeyPad when the virtual keyboard's ESC key is pressed.

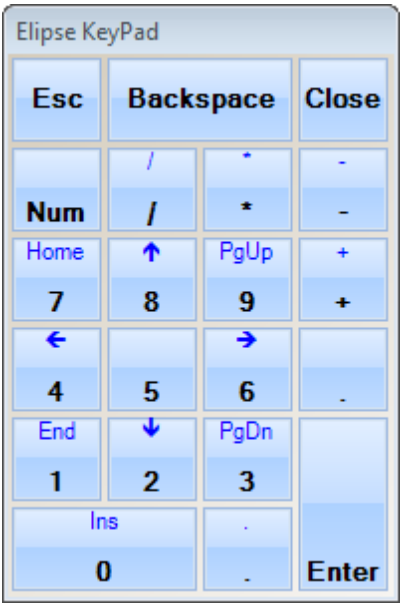
4.3.2.14.2.3 Layout

A Allows changing KeyPad's presentation layout. Possible values for this property are:

- **br-simple**: Displays an alphanumeric keyboard

Example of an alphanumeric KeyPad

- **num**: Displays a numeric keyboard



Example of a numeric KeyPad

Example:

```
' Switches between Alphanumeric and Numeric modes
Sub ToggleButton1_Click()
  If ToggleButton1.Value Then
    Application.GetKeyPad().Layout = "br-simple"
  Else
    Application.GetKeyPad().Layout = "num"
  End If
End Sub
```

4.3.2.14.2.4 SizeFactor

9 Increases or decreases KeyPad's original size, by using a multiplication factor. KeyPad's original size is shown on the next table.

Default size values for KeyPad

LAYOUT	WIDTH	HEIGHT
Alphanumeric	550 px	250 px
Numeric	200 px	300 px

The following example resizes a KeyPad to 75% of its original size.

```
Sub CommandButton1_Click()
  Application.GetKeyPad().SizeFactor = 0.75
  Application.GetKeyPad().Show()
End Sub
```

4.3.2.14.2.5 Sound

A Allows changing a sound played when clicking a KeyPad key. Default value of this property is an empty **String**, meaning that no sound plays when clicking a key. If the value of this property changes, this new value must be an absolute path of a file in WAV format, or this file must be added to a Domain as a Resource.

4.3.2.14.2.6 X

9 The X coordinate of KeyPad's upper left corner. This property can be used to move KeyPad horizontally.

4.3.2.14.2.7 Y

9 The Y coordinate of KeyPad's upper left corner. This property can be used to move KeyPad vertically.

4.4 E3Alarm

This section contains information about **events**, **methods**, and **properties** of the E3Alarm object.

4.4.1 Events

This section contains information about the events of the E3Alarm object.

4.4.1.1 KeyPress

KeyPress(*KeyAscii*)

Occurs when E3Alarm has the keyboard focus and users press a key, which corresponds to a character that can be displayed on screen (an ANSI key, with a code indicated by the *KeyAscii* parameter), that is, this event occurs when one of the following keys is pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- By pressing the TAB key

- By pressing the ENTER key
- By pressing the DEL key (this is not an ANSI key)
- By pressing keyboard arrow keys
- When a key moves the focus from one object to another

While users press a key that produces an ANSI code, E3Alarm receives the **KeyDown** and **KeyPress** events repeatedly. When users release that key, the **KeyUp** event occurs. To monitor keyboard physical status or to handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

NOTE: For a list with all key codes available for the *KeyAscii* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.4.1.2 MouseMove

MouseMove()

Occurs when the mouse pointer is moved over an E3Alarm.

4.4.2 Methods

This section contains information about the methods of the E3Alarm object.

4.4.2.1 AboutBox

AboutBox()

This method displays a dialog box with information about E3Alarm's version and copyright.

4.4.2.2 AckAll

AckAll([*Operator*])

Allows acknowledging all alarms globally. *Operator* is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on E3Alarm's **Operator** column. If omitted, the current Viewer's user is used or "(No User)", if there is no user logged in. For the acknowledgment itself, the logged-in user must have permission to acknowledge alarms.

4.4.2.3 AckCurrentFilter

AckCurrentFilter([*Operator*])

Allows acknowledging all alarms of the current filter. *Operator* is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on E3Alarm's **Operator** column. If omitted, the current Viewer's user is used or "(No User)", if there is no user logged in. For the acknowledgment itself, the logged-in user must have permission to acknowledge alarms.

4.4.2.4 AckSelected

AckSelected([*Operator*])

Allows acknowledging the selected alarms. If there is no selected alarm in E3Alarm, this method fails. Users may acknowledge the alarm (in this case, a new record is inserted into the Database, indicating the acknowledgment), and in E3Alarm the corresponding row indicates that it was acknowledged. *Operator* is an optional **String** indicating the name of the operator that acknowledged the alarm. This value is displayed on E3Alarm's **Operator** column. If omitted, the current Viewer's user is used or "(No User)", if there is no user logged in. For the acknowledgment itself, the logged-in user must have permission to acknowledge alarms.

4.4.2.5 GetEventByIndex

GetEventByIndex(*Index*)

Returns an Event object from a **Collection of Events**, specified by the *Index* parameter, which corresponds to the index of this object on the Collection. For more information about the properties of the object returned by this method, please check topic **Event - Properties**.

4.4.2.6 GetFocusedEvent

GetFocusedEvent()

This method returns an object with all properties of the selected event (the one with current focus) in E3Alarm, if there is a selected event. If no event is selected, this method returns a **Nothing** object.

Properties of the returned object contains field values of the selected event. This object contains a copy of the values at the time of this method's call. Therefore, if the selected event changes, these properties are not updated automatically, and users must use this method whenever updated information about the selected event is needed. Properties of the object returned by this method are described on topic **Event - Properties**.

4.4.3 Properties

This section contains information about the properties of the E3Alarm object.

NOTE: E3 uses, when defining coordinates and widths, the Himetric system. In this system, each logical unit is equivalent to one hundredth of a centimeter, that is, every 1000 units are equivalent to one centimeter. Thus, this is the standard adopted when describing E3 properties, when applicable.

4.4.3.1 ActiveAlarms

9 Determines the total amount of active alarms in an E3Alarm. This is a read-only property.

4.4.3.2 ActiveHighAlarms

9 Indicates the number of active alarms with **High** severity. This is a read-only property.

4.4.3.3 ActiveHighNACKAlarms

9 Indicates the number of unacknowledged alarms with **High** severity. This is a read-only property.

4.4.3.4 ActiveLowAlarms

9 Indicates the number of active alarms with **Low** severity. This is a read-only property.

4.4.3.5 ActiveLowNACKAlarms

9 Indicates the number of unacknowledged alarms with **Low** severity. This is a read-only property.

4.4.3.6 ActiveMedAlarms

9 Indicates the number of active alarms with **Medium** severity. This is a read-only property.

4.4.3.7 ActiveMedNACKAlarms

9 Indicates the number of unacknowledged alarms with **Medium** severity. This is a read-only property.

4.4.3.8 ActiveNACKAlarms

9 Indicates the total amount of unacknowledged alarms in E3Alarm (active or not). This is a read-only property.

4.4.3.9 AlarmCount

9 Determines the number of alarms on an E3Alarm. This is a read-only property.

4.4.3.10 AlarmServer

A Name of a unique Alarm Server in an application.

4.4.3.11 AllowAckAll

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all alarms. Default value of this property is True.

4.4.3.12 AllowAckCurrentFilter

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all alarms of the current filter. If there is no visible alarms, this property has no effect. Default value of this property is True.

4.4.3.13 AllowAckSelected

◩ Enables an E3Alarm's contextual menu option that allows acknowledging all selected alarms. If there is no selected alarms, this property has no effect. Default value of this property is True.

4.4.3.14 AllowColumnClick

◩ Enables or disables field selection and their sort direction, by clicking E3Alarm's column headers at run time. If True, and the header is invisible (please check the **ColumnHeader** property), when users click the column title, data is sorted using this column as the key. When users click the same column title again, the sort order is reversed (from ascending to descending, and vice-versa). When users click it with the SHIFT key pressed, this field is used as a second key. As in the primary key, a second click with SHIFT pressed reverses the sort order of the secondary field.

4.4.3.15 AreaFilter

A Controls visible alarm areas on an E3Alarm. If its value is not an empty **String**, it displays events whose Area names start with the indicated text. For example, if **AreaFilter** is equal to "Ana", it displays Area alarms such as "Analog.Production" or "Analysis", but not "Digital.Analysis" or "Digital.Production". When the

SimpleAreaFilter property is set to True, an Alarm Area also allows using wildcard characters for filtering (such as * or ?) and allows multiple Area filters, separated by colons. Allowed wildcard characters are:

- **"*"**: Accepts none or any amount of characters
- **"?"**: Accepts any character
- **"#"**: Accepts any digit
- **"[ ]"**: Allows specifying a set of characters
  - **"[ab]"**: Accepts a character if it is "a" or "b"
  - **"[f-h]"**: Accepts a character if it is between "f" and "h"
  - **"[!cz]"**: Accepts a character if it is neither "c" nor "z"
  - **"[!m-p]"**: Accepts a character if it is not between "m" and "p"

Default value of this property is an empty **String**, that is, no filtering by area (please check also the **CustomFilter**, **ShowHighPriority**, **ShowMediumPriority**, and **ShowLowPriority** properties).

NOTE: The **AreaFilter** property corresponds to the **Filter** property, available until E3 version 4.0.

4.4.3.16 BannerMode

Enables the visualization of a single message in an E3Alarm. The displayed message depends on the sort configuration and it is always selected. Default value of this property is False. For more information about the sort configuration, please check topic **Sorting Tab**, on **E3 User's Manual**.

4.4.3.17 BorderColor

Defines a color for E3Alarm's border. Default value of this property is black (**RGB(0, 0, 0)**).

4.4.3.18 BorderThickness

Defines a thickness for E3Alarm's border. This property's value ranges from 0 (zero, which disables the border) to 10, and its default value is 1 (one).

4.4.3.19 ColumnHeader

When set to True, this property enables viewing E3Alarm's header. This header also allows performing a table's data reordering visually (please check the **AllowColumnClick** property). Default value of this property is True.

4.4.3.20 Connections

Returns a collection of Connections on an E3Alarm. For more information about the collection returned by this property, please check topic **Collection of Connections**.

4.4.3.21 ConnectionStatusBarColor

Specifies a color for the connection status bar of an E3Alarm. Default value of this property is black (**RGB(0, 0, 0)**).

4.4.3.22 CustomFilter

Allows informing a customized filter for alarms, as an expression. The following fields are available for usage on this filter's expression:

- **Acked (Boolean)**: Indicates whether this message was already acknowledged
- **AckRequired (Boolean)**: Indicates whether this message needs to be acknowledged
- **AckTime (Date)**: Date and time when this alarm's condition was acknowledged (or zero if it was not acknowledged)
- **ActiveSource (Integer)**: **-1 - None**, **0 - ActiveSource**, **1 - Scada**, **2 - Operator**, **3 - CCLink**, **4 - Billing**, **5 - Calculated**, **6 - Database**, **100 - TopologyProcessor**, **101 - PowerFlow**, **102 - StateEstimator**, **103 - LoadShedding**, **104 - DistLoadModelling**, **105 - SelfHealing**, or **106 - ExternalReader**
- **ActorID (String)**: Login of the user that acknowledged this message (or an empty **String** if this message was not yet acknowledged)
- **AlarmSourceName (String)**: Name of the Alarm Source object (only the name, not its full path)
- **Area (String)**: Area of this alarm
- **ChangeMask (Integer)**: Field currently not used by E3, it is always 0 (zero)
- **ConditionActive (Boolean)**: Indicates whether the alarm condition is active
- **ConditionName (String)**: Name of the last active alarm condition
- **Cookie (Integer)**: Identifies an Alarm Source during an execution session
- **CurrentValue (Double)**: Value of the Source at the moment the alarm condition became active
- **Enabled (Boolean)**: Indicates whether an alarm verification on the Alarm Source is enabled
- **EventCategory (String)**: Name of the alarm category (for example, "Level",

"Rate of Change", "Dead Band", "Digital", or "Discrete")

- **EventTime (Date)**: Date and time of the last event update
- **EventTimeUTC (Date)**: Date and time of the last event update
- **EventType (String)**: "Event" (event) or "Condition" (alarm)
- **FormattedValue (String)**: Contains the Source's value (formatted) at the moment the alarm condition became active
- **FullAlarmSourceName (String)**: Full name of the Alarm Source object
- **InTime (Date)**: Date and time the alarm condition became active
- **Message (String)**: Alarm's message
- **OutTime (Date)**: Date and time the condition left the alarm (or zero if it is still active)
- **Quality (String)**: "Good (xxx)", "Bad (xxx)", or "Uncertain (xxx)"
- **Severity (Integer)**: **0 - High**, **1 - Medium**, or **2 - Low**
- **Source (String)**: Link of the Alarm Source
- **SubConditionName (String)**: Name of the alarm's sub-condition (for example, "LOLO", "LO", "HI", "HIHI", "DIG", etc.)

User-defined fields can also be used on a filter expression, by using the name defined on the Alarm Server.

Altogether, messages displayed on E3Alarm list always pass through five filters:

- Filter by type (alarm or event) (the **FilterType** property)
- Filter by severity (the **ShowLowPriority**, **ShowMediumPriority**, and **ShowHighPriority** properties)
- Filter by area (the **AreaFilter** and **SimpleAreaFilter** properties)
- Filter by the **CustomFilter** property
- Filter by the alarm summary (equivalent to the expression "Enabled AND (ConditionActive OR (AckRequired AND NOT Acked))")

Usage examples of the **CustomFilter** property:

- For a user-defined field called **IsSupressed**, and that users want to display only alarms where this field's value is different from zero, the expression to use is the following:

```
IsSupressed <> 0
```

- To display only messages with the sub-condition "HIHI" or "LOLO" of alarm objects whose name starts with "Pressure", the expression to use is the following:

```
(SubConditionName = "HIHI" OR SubConditionName = "LOLO")
  AND (Mid(AlarmSourceName, 1, 8) = "Pressure")
```

4.4.3.23 Domain

A Specifies the Domain to which an E3Alarm connects. Default value is an empty **String**, that is, an E3Alarm connects to the same Viewer's Domain where it is. For example, "\\NameOfAnotherServer".

4.4.3.24 Events

Returns a collection of events in an E3Alarm. For more information about the collection returned by this property, please check topic **Collection of Events**.

4.4.3.25 Filters

Returns a collection of Filters on an E3Alarm. For more information about the collection returned by this property, please check topic **Collection of Filters**.

4.4.3.26 FilterType

Performs the alarm filters. The available options are the following:

- **1 - OnlyAlarms**: Displays only alarms
- **2 - OnlyEvents**: Displays only events
- **3 - AlarmsAndEvents**: Displays both alarms and events

4.4.3.27 Font

A Determines the font for E3Alarm's header and rows. This is a read-only property and can only be modified in Studio, not at run time.

4.4.3.28 FourthSortAscending

When this property is set to False, the sort order of events by a fourth field is performed in descending order. Otherwise, it is performed in ascending order. Default value of this property is False.

4.4.3.29 FourthSortField

A Determines a fourth field for sorting events in an E3Alarm. The name of this field must be always specified in English (please check all available fields in **E3 User's Manual**). Default value of this property is an empty **String**. This property has no effect when the **PrimarySortField**, **SecondarySortField**, or **ThirdSortField** properties

are configured as an empty **String**.

4.4.3.30 GridBkColor

Determines a color for E3Alarm's background. Default value of this property is the color configured in Windows for the **Window** item (**Control Panel - Display - Appearance - Advanced**).

4.4.3.31 InactiveHighNACKAlarms

Indicates the number of inactive and unacknowledged alarms with **High** severity. This is a read-only property.

4.4.3.32 InactiveLowNACKAlarms

Indicates the number of inactive and unacknowledged alarms with **Low** severity. This is a read-only property.

4.4.3.33 InactiveMedNACKAlarms

Indicates the number of inactive and unacknowledged alarms with **Medium** severity. This is a read-only property.

4.4.3.34 InactiveNACKAlarms

Determines the total amount of inactive and unacknowledged alarms. This is a read-only property.

4.4.3.35 PictureConnected

Path of an image file with an icon to represent a Connection from E3Alarm's Collection of Connections successfully connected to an Alarm Server. Default value for this property is an empty **String** (an E3Alarm displays the icon on its connection status bar for this Connection).

4.4.3.36 PictureNotConnected

Path of an image file with an icon to represent a Connection from E3Alarm's Collection of Connections not connected to an Alarm Server. Default value for this property is an empty **String** (an E3Alarm displays the icon on its connection status bar for this Connection).

4.4.3.37 PictureUnknown

Path of an image file with an icon to represent a Connection on E3Alarm's Collection of Connections with an unknown status. Default value for this property is

an empty **String** (an E3Alarm displays the icon on its connection status bar for this Connection).

NOTE: An unknown status indicates that the E3Alarm did not receive a confirmation from the Alarm Server that this connection was successful. This is the standard status for versions earlier than 4.7.

4.4.3.38 PopupMenu

◩ Enables a contextual menu when right-clicking an E3Alarm object. Default value of this property is True.

4.4.3.39 PrimarySortAscending

◩ When this property is set to False, the sort order of events by a primary field is performed in descending order. Otherwise, it is performed in ascending order. The default value of this property is False.

4.4.3.40 PrimarySortField

A Determines a primary field for sorting events in an E3Alarm. The name of this field must always be specified in English (please check all available fields in **E3 User's Manual**). Default value of this property is "EventTime". When this property is an empty **String**, the **SecondarySortField**, **ThirdSortField**, and **FourthSortField** properties do not have any effect.

4.4.3.41 SecondarySortAscending

◩ When this property is set to False, the sort order of events by a secondary field is performed in descending order. Otherwise, it is performed in ascending order. Default value of this property is False.

4.4.3.42 SecondarySortField

A Determines a secondary field for sorting events in an E3Alarm. The name of this field must always be specified in English (please check all available fields in **E3 User's Manual**). Default value of this property is an empty **String**. This property does not have any effect when the **PrimarySortField** property is an empty **String**.

4.4.3.43 ShowConnectionStatusBar

◩ Shows or hides E3Alarm's connection status bar. Default value for this property is False.

4.4.3.44 ShowHighPriority

◩ Filters which alarms are displayed, according to their severity. When this property is set to True, alarms with **High** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.3.45 ShowLowPriority

◩ Filters which alarms to display, according to their severity. When this property is set to True, alarms with **Low** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.3.46 ShowMediumPriority

◩ Filters which alarms to display, according to their severity. If this property is set to True, alarms with **Medium** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.3.47 SimpleAreaFilter

◩ When this property is set to True, the behavior of filtering by Alarm Area names is based only on the coincidence of the initial part of a name. When this property is set to False, this behavior considers the entire Area name, but allows using wildcard characters and multiple area filters, which must be separated by colons. Please check also the **AreaFilter** property, which specifies a filter by area name.

4.4.3.48 ThirdSortAscending

◩ When this property is set to False, the sort order of events by a third field is performed in descending order. Otherwise, it is performed in ascending order. Default value of this property is False.

4.4.3.49 ThirdSortField

A Determines a third field for sorting events in an E3Alarm. The name of this field must always be specified in English (please check all available fields in **E3 User's Manual**). Default value of this property is an empty **String**. This property has no effect when the **PrimarySortField** or **SecondarySortField** properties are configured as an empty **String**.

4.4.4 Collection of Connections

This section contains information about **methods** and **properties** of the Collection of Connections object. This object does not have events associated to it.

NOTE: The Collection of Connections can be accessed by using the **Connections** property of **Alarm Filter** and **E3Alarm** objects.

4.4.4.1 Methods

This section contains information about the methods of the Collection of Connections object.

4.4.4.1.1 Add

Add(*ConnectionName*)

Adds a new Connection, with the name informed in the *ConnectionName* parameter and returns this Connection. If users try to create a Connection with an already existing name, an error message is then displayed. To generate a name automatically, leave the *ConnectionName* parameter empty.

4.4.4.1.2 Item

Item(*Index*)

Returns a Connection object from the Collection of Connections, specified by the *Index* parameter. This parameter can be a number, if it corresponds to the object's index on the Collection, or a text if it corresponds to the Connection's name (the Connection's **ConnectionName** property).

4.4.4.1.3 Remove

Remove(*Index*)

Removes a Connection object from the Collection of Connections by its name or index, specified by the *Index* parameter. The Connection with index 0 (zero) cannot be removed. If users try to remove it, an error message is then displayed.

4.4.4.2 Properties

This section contains information about the properties of the Collection of Connections object.

4.4.4.2.1 Count

9 Contains the total amount of Connections on the Collection of Connections. This is a read-only property.

4.4.4.3 Connection

This section contains information about **properties** of the Connection object from a Collection of Connections. This object does not have events nor methods associated

to it.

4.4.4.3.1 Properties

This section contains information about the properties of the Connection object from a Collection of Connections.

4.4.4.3.1.1 AlarmServer

A Name of the Alarm Server, which can be located on a local Domain or on a remote Domain.

4.4.4.3.1.2 ConnectionName

A Name that identifies this Connection.

4.4.4.3.1.3 ConnectionStatus

Current status of this Connection. This is a read-only property. Possible values for this property are the following:

- **0 - Not connected**: It is not connected to the selected Alarm Server
- **1 - Connected**: It is connected to the selected Alarm Server
- **2 - Filter not configured**: The Connection filter (the **FilterConnection** property) is not properly configured
- **3 - Alarm Server not configured**: The Alarm Server (the **AlarmServer** property) is not configured
- **4 - Unexpected error**: There was a unexpected error on this Connection
- **5 - Unidentified status**: A confirmation for a connection with the Alarm Server was not received. This Alarm Server is possibly on a remote Domain with an E3 version earlier than 4.7

4.4.4.3.1.4 DomainName

A Friendly name to identify a Domain where the selected Alarm Server is located. This name is displayed on E3Alarm's **Source Domain** column.

4.4.4.3.1.5 FilterConnection

A Name of a **Filter** object that contains a filter configuration for alarms, available on the **Collection of Filters**.

4.4.5 Collection of Events

This section contains information about **methods** and **properties** of the Collection of Events object. This object does not have events associated to it.

NOTE: The Collection of Events can be accessed by using the **Events** property of **Alarm Filter** and **E3Alarm** objects.

4.4.5.1 Methods

This section contains information about the methods of the Collection of Events object.

4.4.5.1.1 Item

Item(*Index*)

Returns an Event object from a Collection of Events, specified by the *Index* parameter. This parameter can be a number, if it corresponds to the index of this object on that Collection, or a text, if it corresponds to Alarm Source's name (Event's **AlarmSourceName** property).

4.4.5.2 Properties

This section contains information about the properties of the Collection of Events object.

4.4.5.2.1 Count

9 Contains the total amount of Events on a Collection of Events. This is a read-only property.

4.4.5.3 Event

This section contains information about **properties** of the Event object from a Collection of Events. This object does not have events nor methods associated to it.

4.4.5.3.1 Properties

This section contains information about the properties of the Event object from a Collection of Events.

4.4.5.3.1.1 Acked

◩ Informs whether this event was acknowledged (True) or not (False). This is a read-only property.

4.4.5.3.1.2 AckRequired

◩ Determines an automatic (False) or manual (True) acknowledgment of this event. This is a read-only property.

4.4.5.3.1.3 AckTime

Registers E3's date and time at the moment this event is acknowledged, or zero (12/30/1899) while this event is not acknowledged. For events that do no ask for acknowledgment, this property assumes E3's date and time at the moment this event is activated. This is a read-only property.

4.4.5.3.1.4 ActiveSource

9 Active Measurement Source of the Alarm Source at the moment of this event. This is a read-only property.

4.4.5.3.1.5 ActorID

A Name of the operator that acknowledged this event. This is a read-only property and its value can be:

- Viewer's logged-in user, when acknowledgment is performed on E3Alarm, or "No user" if there is no logged-in user
- "System", when acknowledgment is automatic, that is, for events that ask for acknowledgment
- A name passed by script, such as when using Alarm Server's **AckAllAlarms**, **AckArea**, or Alarm Source's **Ack** method

4.4.5.3.1.6 AlarmSourceName

A Contains the name of the Alarm Source. This is a read-only property.

4.4.5.3.1.7 Area

A For alarm events, this is the name of the Area to which the Alarm Source belongs. This is a read-only property.

4.4.5.3.1.8 ConditionActive

◩ Indicates whether the Alarm Source is in alarm (True) or not (False). This is a read-only property.

4.4.5.3.1.9 ConditionName

A Name of the condition, if this is an alarm event. This property can have the following values:

- **Dead Band**: **Dead Band**-type Alarm Source
- **Digital**: **Digital**-type Alarm Source
- **Discrete**: **Discrete**-type Alarm Source
- **Level**: **Analog**-type Alarm Source
- **RateOfChange**: **Rate of Change**-type Alarm Source

If this event is not an alarm this property is always an empty **String**. This is a read-only property.

4.4.5.3.1.10 CurrentValue

9 Determines the value of the Alarm Source (converted to **Double**) at the moment of this event. For other events this property's value is always 0 (zero). This is a read-only property.

4.4.5.3.1.11 DomainName

A Returns the domain name of this event's Connection. This name corresponds to the **Domain** column on the list of available Connections on E3Alarm's **Connections** tab. This is a read-only property.

4.4.5.3.1.12 Enabled

◩ Determines whether alarm check is enabled (True) or not (False). This is a read-only property.

4.4.5.3.1.13 EventCategory

A Event's category. For alarms, this property can have the following values:

- **Dead Band**: **Dead Band**-type Alarm Source
- **Digital**: **Digital**-type Alarm Source
- **Discrete**: **Discrete**-type Alarm Source
- **Level**: **Analog**-type Alarm Source
- **RateOfChange**: **Rate of Change**-type Alarm Source

This is a read-only property.

4.4.5.3.1.14 EventCLSID

A Unique identifier for this event's lifespan. When a new event occurs in a Source, a new EventCLSID is generated. Thus, this event keeps this CLSID on the database, while it does not leave the list of active and not acknowledged events. This is a read-only property.

4.4.5.3.1.15 EventTime

Date and time of Alarm Source's value at the moment of this event. This is a read-only property.

4.4.5.3.1.16 EventTimeUTC

Date and time of Alarm Source's value at the moment of this event, relative to Greenwich Mean Time. This value is the same as the **EventTime** property, and it is kept in E3 for compatibility reasons. This is a read-only property.

4.4.5.3.1.17 EventType

A Event's type. For alarm events, this is always "Condition". This is a read-only property.

4.4.5.3.1.18 FormattedValue

A Displays the formatted value of the Alarm Source that goes to this event. This is a read-only property.

4.4.5.3.1.19 FullAlarmSourceName

A Registers the Alarm Source's full path, including Areas, name of Alarm Configurations, and possible Folders where it can be inserted. For example, "Folder1.AlarmConfig1.Area1.AlarmSource1". This is a read-only property.

4.4.5.3.1.20 InTime

Registers value's date and time at the moment when it enters the alarm condition. This is a read-only property.

4.4.5.3.1.21 Message

A This is the text configured on the Alarm Source. This is a read-only property.

4.4.5.3.1.22 OutTime

Registers value's date and time at the moment in which it leaves the alarm condition, or zero (12/30/1899) while this alarm still did not leave the active condition. This is a read-only property.

4.4.5.3.1.23 Quality

Quality of Alarm Source's value at the moment of this event. This is a read-only property and it can assume the following numerical values:

- **0 - 63**: Bad quality
- **64 - 127**: Uncertain quality
- **128 - 191**: Undefined value
- **192 - 255**: Good quality

If this event is not an alarm this property is equal to an empty **String**. Example: "Bad (0)", "Uncertain (64)", "?? (128)", or "Good (192)".

4.4.5.3.1.24 Severity

This is the value of severity configured on the Alarm Source. This property can assume values **0: High**, **1: Medium**, or **2: Low**. This is a read-only property.

4.4.5.3.1.25 Source

For alarm events, informs the expression used to evaluate alarm conditions. This is a read-only property.

4.4.5.3.1.26 SubConditionName

Name of the subcondition, if it is an alarm event. This property can assume the following values:

- **DB**: Dead Band alarm
- **DIG**: Digital alarm
- **ROC**: Rate of Change alarm
- **LOLO**: Analog alarm on a **Very Low** range
- **LO**: Analog alarm on a **Low** range
- **HI**: Analog alarm on a **High** range
- **HIHI**: Analog alarm on a **Very High** range

NOTE: For **Discrete**-type Alarm Sources, this property assumes the name of the user-defined Subcondition (the **Discrete** tab of properties of this type of Alarm).

If this event is not an alarm this property is equal to an empty **String**. This is a read-only property.

4.4.5.3.1.27 UserField

9 This property receives the index of the Collection of User Fields of the Alarm Area (the **UserFields** property) and returns its corresponding object. For more information about the object returned by this property, please check topic **Alarm's User Fields**. This is a read-only property.

4.4.6 Collection of Filters

This section contains information about **methods** and **properties** of the Collection of Filters object. This object does not have events associated to it.

NOTE: The Collection of Filters can be accessed by using the **Filters** property of **Alarm Filter** and **E3Alarm** objects.

4.4.6.1 Methods

This section contains information about the methods of the Collection of Filters object.

4.4.6.1.1 Add

Add(*FilterName*)

Adds a new Filter with the name informed in the *FilterName* parameter and returns this Filter. If users try to create a Filter with an already existing name, an error message is then displayed. To generate a name automatically, leave the *FilterName* empty.

4.4.6.1.2 Item

Item(*Index*)

Returns a Filter object from the Collection of Filters, specified by the Index parameter. This parameter can be a number, if it corresponds to the object's index on the Collection, or a text if it corresponds to the Filter's name (the Filter's **FilterName** property).

4.4.6.1.3 Remove

Remove(*Index*)

Removes a Filter object from the Collection of Filters by its name or index, specified by the *Index* parameter. The Filter with index 0 (zero) cannot be removed. If users try to remove it, an error message is then displayed.

4.4.6.2 Properties

This section contains information about the properties of the Collection of Filters object.

4.4.6.2.1 Count

9 Contains the total amount of Filters on the Collection of Filters. This is a read-only property.

4.4.6.3 Filter

This section contains information about **properties** of the Filter object from a Collection of Filters. This object does not have events nor methods associated to it.

4.4.6.3.1 Properties

This section contains information about the properties of the Filter object from a Collection of Filters.

4.4.6.3.1.1 AreaFilter

A Controls visible alarm areas on a Filter. If its value is not an empty **String**, it displays events whose Area names start with the indicated text. For example, if **AreaFilter** is equal to "Ana", it displays Area alarms such as "Analog.Production" or "Analysis", but not "Digital.Analysis" ou "Digital.Production". When the **SimpleAreaFilter** property is set to True, an Alarm Area also allows using wildcard characters for filtering (such as * or ?) and allows multiple Area filters, separated by colons. Allowed wildcard characters are:

- **"*"**: Accepts none or any amount of characters
- **"?"**: Accepts any character
- **"#"**: Accepts any digit
- **"[ ]"**: Allows specifying a set of characters
  - **"[ab]"**: Accepts a character if it is "a" or "b"
  - **"[f-h]"**: Accepts a character if it is between "f" and "h"

- **"[!cz]"**: Accepts a character if it is neither "c" nor "z"
- **"[!m-p]"**: Accepts a character if it is not between "m" and "p"

Default value of this property is an empty **String**, that is, no filtering by area (please check also the **CustomFilter**, **SimpleAreaFilter**, **ShowHighPriority**, **ShowMediumPriority**, and **ShowLowPriority** properties).

4.4.6.3.1.2 CustomFilter

A Allows informing a customized filter for alarms, as an expression. The following fields are available for usage on this filter's expression:

- **Acked (Boolean)**: Indicates whether this message was already acknowledged
- **AckRequired (Boolean)**: Indicates whether this message needs to be acknowledged
- **AckTime (Date)**: Date and time when this alarm's condition was acknowledged (or zero if it was not acknowledged)
- **ActiveSource (Integer)**: **-1: None**, **0: ActiveSource**, **1: Scada**, **2: Operator**, **3: CCLink**, **4: Billing**, **5: Calculated**, **6: Database**, **100: TopologyProcessor**, **101: PowerFlow**, **102: StateEstimator**, or **103: LoadShedding**
- **ActorID (String)**: Login of the user that acknowledged this message (or an empty **String** if this message was not yet acknowledged)
- **AlarmSourceName (String)**: Name of the Alarm Source object (only the name, not its full path)
- **Area (String)**: Area of this alarm
- **ChangeMask (Integer)**: Field currently not used by E3, it is always 0 (zero)
- **ConditionActive (Boolean)**: Indicates whether the alarm condition is active
- **ConditionName (String)**: Name of the last active alarm condition
- **Cookie (Integer)**: Identifies an Alarm Source during an execution session
- **CurrentValue (Double)**: Value of the Source at the moment the alarm condition became active
- **Enabled (Boolean)**: Indicates whether an alarm check on the Alarm Source is enabled
- **EventCategory (String)**: Name of the alarm category (for example, "Level", "Rate of Change", "Dead Band", "Digital", or "Discrete")
- **EventTime (Date)**: Date and time of the last event update
- **EventTimeUTC (Date)**: Date and time (UTC) of the last event update

- **EventType (String)**: "Event" (event) or "Condition" (alarm)
- **FormattedValue (String)**: Contains the Source's value (formatted) at the moment the alarm condition became active
- **FullAlarmSourceName (String)**: Full name of the Alarm Source object
- **InTime (Date)**: Date and time the alarm condition became active
- **Message (String)**: Alarm's message
- **OutTime (Date)**: Date and time the condition left the alarm (or zero if it is still active)
- **Quality (String)**: "Good (xxx)", "Bad (xxx)", or "Uncertain (xxx)"
- **Severity (Integer)**: **0: High**, **1: Medium**, or **2: Low**
- **Source (String)**: Link of the Alarm Source
- **SubConditionName (String)**: Name of the alarm's sub-condition (for example, "LOLO", "LO", "HI", "HIHI", "DIG", etc.)

User-defined fields can also be used on a filter expression, by using the name defined on the Alarm Server.

Altogether, messages displayed on a Filter list always pass through five filters:

- Filter by type (alarm or event) (the **FilterType** property)
- Filter by severity (the **ShowLowPriority**, **ShowMediumPriority**, and **ShowHighPriority** properties)
- Filter by area (the **AreaFilter** and **SimpleAreaFilter** properties)
- Filter by the **CustomFilter** property
- Filter by the alarm summary (equivalent to the expression "Enabled AND (ConditionActive OR (AckRequired AND NOT Acked))")

For usage examples of this property, please check E3Alarm's **CustomFilter** property. Default value of this property is an empty **String**.

4.4.6.3.1.3 FilterName

A Name that identifies this Filter.

4.4.6.3.1.4 FilterType

Performs the alarm filters. Available options are the following:

- **1 - OnlyAlarms**: Displays only alarms

- **2 - OnlyEvents**: Displays only events
- **3 - AlarmsAndEvents**: Displays both alarms and events

Default value of this property is **1 - OnlyAlarms**.

4.4.6.3.1.5 ShowHighPriority

◩ Filters which alarms are displayed, according to their severity. When this property is set to True, alarms with **High** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.6.3.1.6 ShowLowPriority

◩ Filters which alarms are display, according to their severity. When this property is set to True, alarms with **Low** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.6.3.1.7 ShowMediumPriority

◩ Filters which alarms to display, according to their severity. If this property is set to True, alarms with **Medium** severity are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

4.4.6.3.1.8 SimpleAreaFilter

◩ When this property is set to True, the behavior for filtering by Alarm Area names is based only on the coincidence of the initial part of a name. When this property is set to False, this behavior considers the entire Area name, but allows using wildcard characters and multiple area filters, which must be separated by colons. Please check also the **AreaFilter** property, which specifies a filter by area name. Default value of this property is True.

4.5 E3Browser

This section contains information about **events**, **methods**, and **properties** of the E3Browser object.

4.5.1 Events

This section contains information about the events of the E3Browser object.

4.5.1.1 KeyPress

KeyPress(*KeyAscii*)

This event occurs when the object has the keyboard focus, and the user presses a

key corresponding to a character that can be showed on Screen (an ANSI key, with a code indicated by the *KeyAscii* parameter). That is, the event occurs when some of the following keys are pressed:

- Any keyboard character that can be printed
- The CTRL key combined with any standard alphabet character
- The CTRL key combined with any special character
- The BACKSPACE key
- The ESC key

This event does not occur in the following conditions:

- By pressing the TAB key
- By pressing the ENTER key
- By pressing the DEL key (this is not an ANSI key)
- By pressing keyboard arrow keys
- When a key change the focus from one object to another

While a user presses a key that produces an ANSI code, the object repeatedly receives the **KeyDown** and the **KeyPress** events. When the user releases the key, the **KeyUp** event occurs.

To monitor keyboard physical status, or handle keys not recognized by the **KeyPress** event (such as function keys, browsing keys, etc.), use the **KeyDown** and **KeyUp** events.

NOTE: For a list with all key codes available for the *KeyAscii* parameter, please check the article *Keys Enumeration* on **Microsoft Developer Network**.

4.5.1.2 OnDrawRow

OnDrawRow(*Selected*, *Row*, *TextColor*, *BackColor*)

This method passes four parameters. The *Selected* parameter indicates whether the row is selected. The *Row* parameter indicates the number of the row being drawn. The *TextColor* parameter indicates the row's text color, and the *BackColor* parameter indicates the text's background color. If the color is modified within this event, this change is used by the E3Browser when drawing the row. Another important new feature is that if the **GetColumnValue** method is called within this event, returned values are from the drawn row, not the selected row.

4.5.1.3 OnFormatCell

OnFormatCell(*Column*, *FieldName*, *OriginalValue*, *FormattedValue*)

This event allows customizing the format of E3Browser's cells text. Parameters of this event are the following:

- **Column**: Index of E3Browser's visible column (starting at 0). It allows identifying the cell's column being formatted
- **FieldName**: A text with the name of the column's field being formatted
- **OriginalValue**: Cell's unformatted **Variant**-type value
- **FormattedValue**: A formatted **Variant**-type value, according to the configuration of E3Browser's column. If it is modified inside the event, then it allows changing the formatted value

Example (formatting Alarm fields):

```
Sub E3Browser1_OnFormatCell(Column, FieldName, OriginalValue,
FormattedValue)
  If Column = 15 Then
    If Not IsNull(OriginalValue) Then
      FormattedValue = SourceTypeName(OriginalValue)
  ElseIf Column = 9 Then
    If OriginalValue = 0 Then
      FormattedValue = "High"
    ElseIf OriginalValue = 1 Then
      FormattedValue = "Medium"
    Else
      FormattedValue = "Low"
    End If
  End If
End Sub
```

4.5.1.4 MouseMove

MouseMove()

Occurs when the mouse pointer is moved over the E3Browser.

4.5.2 Methods

This section contains information about the methods of the E3Browser object.

4.5.2.1 AboutBox

AboutBox()

This method displays a dialog box with information about E3Browser's version and

copyright.

4.5.2.2 ClearFields

ClearFields()

Clears E3Browser's column and row format.

4.5.2.3 GetColumnValue

GetColumnValue(*Index*)

Returns a cell value, in the informed column and selected row. This method has the *Index* parameter, determining the index of the desired column. Example:

```
Sub E3Browser1_DblClick()
  Screen.Item("Text1").Value = GetColumnValue(0)
  Screen.Item("Text2").Value = GetColumnValue(1)
  Screen.Item("Text3").Value = GetColumnValue(2)
  Screen.Item("Text4").Value = GetColumnValue(3)
End Sub
```

4.5.2.4 Requery

Requery()

The **Requery** method updates the Query, by using its current settings, and then returns data to the E3Browser.

4.5.2.5 RetrieveE3QueryFields

RetrieveE3QueryFields()

The **RetrieveE3QueryFields** method reads a Query data structure and updates the E3Browser format with the fields defined by that Query. If it is successful, it returns True. Otherwise, it returns False. This method is especially useful when users must use a single E3Browser to display data from different tables or queries.

4.5.3 Properties

This section contains information about the properties of the E3Browser object.

4.5.3.1 AllowColumnResize

◩ Enables or disables E3Browser's grid column size configuration, at run time. If False, column size is fixed, and cannot be modified.

4.5.3.2 AllowRowResize

Enables or disables E3Browser's grid row size configuration, at run time. If False, row size is fixed, and cannot be modified.

4.5.3.3 ColumnWidth

Determines E3Browser's column width, in pixels.

4.5.3.4 CurSel

Indicates E3Browser cursor's current position, that is, the row index where cursor is positioned.

4.5.3.5 E3Query

Returns E3Browser's Query object, so that its properties can be accessed.

4.5.3.6 Fields

Returns the Collection that contains a list of all table fields, allowing its reference through the items of this collection. Default value is empty. Example:

```
Sub E3Browser1_Click()
  ' Changes the color of Field1
  Set fields = Screen.Item("E3Browser").Fields
  Set Field1 = fields.Item("Field1")
  field1.BkColor = RGB(255, 0, 0) ' Red
  ' Shows how many fields E3Browser has
  MsgBox fields.Count
  ' Shows E3Browser's number of fields
  For Each Fields In fields
    MsgBox fields.Name
  Next
End Sub
```

4.5.3.7 FixedBkColor

Specifies E3Browser's first column background color. Default value is beige (**RGB(236, 233, 216)**).

4.5.3.8 FixedColumnWidth

Specifies E3Browser's first column width (in pixels). Default value is 30.

4.5.3.9 FixedRowFont

Determines the font to be used at E3Browser's header row. This property cannot be used in scripts or Links, and is configured only via Studio. Default value is "Arial".

4.5.3.10 FixedRowHeight

Determines E3Browser's header row height (in pixels). Default value is 20.

4.5.3.11 FixedTextColor

Changes E3Browser's header color.

4.5.3.12 GridBkColor

Determines E3Browser's data area background color. Default value is white (**RGB(255, 255, 255)**).

4.5.3.13 GridFont

Determines the font to be used on E3Browser's data area texts. Default value is "Arial". This property cannot be used in scripts or Links, being configured only via Studio.

4.5.3.14 GridLineColor

Determines E3Browser's data grid line color. Default value is gray (**RGB(192, 192, 192)**).

4.5.3.15 GridLinesType

Determines the type of rows to be drawn on E3Browser's data grid.

Available options for GridLinesType

OPTION	DESCRIPTION
0 - GLNone	No grid lines.
1 - GLHorz	Only horizontal grid lines (default).
2 - GLVert	Only vertical grid lines.
3 - GLBoth	Horizontal and vertical grid lines.

4.5.3.16 RefreshTime

Specifies Query's refresh time, relative to the specific database. Through this property it is possible to verify data update at the related Historic referring to a

specific time (in milliseconds). When the **RefreshTime** property is equal to 0, there is no data update, therefore data remains unchanged.

4.5.3.17 RowHeight

9 Defines E3Browser's rows height, in pixels. Default value is 20.

4.5.3.18 SelectRow

◩ Determines whether it is possible to select E3Browser's rows. If True, these rows can be selected. Otherwise, row selection is disabled.

4.5.3.19 SourceQuery

A Contains a reference to an E3Query object connected to this E3Browser.

NOTE: To change E3Browser's Query via scripts (in case the new Query modifies the original Query fields), in addition to changing the **SourceQuery** property, users must use the **RetrieveE3QueryFields** and **Requery** methods.

4.5.3.20 TextBkColor

Specifies E3Browser's data cells background color. Default value is white (**RGB(255, 255, 255)**).

4.5.3.21 TextColor

Specifies E3Browser's text color. Default value is black (**RGB(0, 0, 0)**).

4.5.3.22 TitleTipBkColor

Specifies E3Browser's tip background color. Default value is black (**RGB(0, 0, 0)**).

4.5.3.23 TitleTipTextColor

Specifies E3Browser's tip text color. Default value is gray (**RGB(204, 204, 204)**).

4.5.3.24 ToolbarBkColor

Specifies E3Browser's toolbar background color. Default value is beige (**RGB(236, 233, 216)**).

4.5.3.25 ToolbarFont

A Determines E3Browser's toolbar text font. This property cannot be used in scripts or Links, being configured only via Studio.

4.5.3.26 ToolbarForeColor

Specifies E3Browser's toolbar foreground color. Default value is black (**RGB(0, 0, 0)**).

4.5.4 E3Browser Fields

This section contains information about **properties** of the E3Browser Field object. This object does not have events nor methods associated to it.

4.5.4.1 Properties

This section contains information about the properties of the E3Browser field object.

4.5.4.1.1 BkColor

Determines E3Browser field's background color. The default value is the color configured on Windows for the **Window** item on **Control Panel (Control Panel - Display - Appearance - Advanced)**.

4.5.4.1.2 Color

Returns the Field's text color. The default value of this property is black (**RGB(0, 0, 0)**).

4.5.4.1.3 Format

Configures the formatter used on the Field column.

4.5.4.1.4 Name

Returns the Field's name.

4.5.4.1.5 Visible

Enables or disables the visibility of the selected Field on E3Browser's Query. If this property is configured to True, this Field is visible on E3Browser. Otherwise, this Field is not shown on E3Browser at run time. The default value is True.

4.5.4.1.6 Width

Returns the Field's width, in pixels.

4.6 E3Chart

This section contains information about **events**, **methods**, and **properties** of the E3Chart object.

4.6.1 Events

This section contains information about the events of the E3Chart object.

4.6.1.1 OnCursorChange

OnCursorChange()

Occurs whenever an E3Chart cursor changes its position. For example, users can create a script for this event when they need to show cursor position values on a Screen. Example:

```
Sub E3Chart1_OnCursorChange()
  Set Chart = Application.GetFrame("").Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.Item(0)
  ' The Text1 object must display
  ' cursor's current position
  Set Text = Application.GetFrame("").Screen.Item("Text1")
  If Pen.GetCursorPos(aa, bb) Then
    Text.Value = "X Position = " & aa & "; Y Position = " & bb
  End If
End Sub
```

4.6.1.2 OnLegendClick

OnLegendClick(*Row*, *Col*, *RowData*)

Occurs whenever users click one of the Legend rows. The *Row* and *Col* parameters indicate, respectively, the row and column clicked. The *RowData* parameter is the Legend's Pen index where the click occurred. Example:

```
Sub E3Chart1_OnLegendClick(Row, Col, RowData)
  Set text = Screen.Item("Text1")
  text.Value = Legend.Item(col).Name & " " & _
    Pens.Item(RowData).name
End Sub
```

4.6.1.3 OnQueryFinish

OnQueryFinish()

Occurs whenever one or more Queries are finished. When this event is generated, a call for the **FitAll** or **FitPen** methods may cause problems if users are using automatic Queries, since these methods activate other Queries until all data is read. In this case, it is recommended that the value passed by these methods'

parameters be 1 (one), which fits all Pens vertically.

4.6.2 Methods

This section contains information about the methods of the E3Chart object.

4.6.2.1 ClearPenMarks

ClearPenMarks()

Removes all search marks from all E3Chart Pens.

4.6.2.2 CopyConfig

CopyConfig(*SourceChart*[, *Flags*])

The **CopyConfig** method copies settings from an E3Chart to another one. The *SourceChart* parameter indicates the source E3Chart, whose properties are copied to the E3Chart calling this method.

NOTE: In case of Reports, the **CopyConfig** method works only with **Historic**-type Pens.

For example, to copy settings from an E3Chart on a Screen (`ScreenChart`) to another one within a Report (`ReportChart`), the following script must be added to the Report object associated to the Report.

```
Sub OnBeforePrint
  Set Chart = _
    Report.Sections("PageHeader").Controls("ReportChart")
  Chart.CopyConfig(Application.GetFrame()._
    Screen.Item("ScreenChart"))
  Chart.LoadData()
  Chart.FitAll()
End Sub
```

NOTE: This method also has an optional and not used parameter called *Flags*, kept only for compatibility reasons.

4.6.2.3 FitAll

FitAll([*FitStyle*])

Fits all Pens into an E3Chart. The optional parameter *FitStyle* indicates the way Pens fit at run time:

- **0**: Fits both Axes at the same time
- **1**: Fits only the Vertical Axis

- **2**: Fits only the Horizontal Axis

4.6.2.4 FitPen

FitPen(*Pen*[, *FitStyle*])

Fits a Pen into an E3Chart specified by index or name. The *Pen* parameter defines the Pen to fit into the E3Chart. The optional parameter *FitStyle* indicates how Pens fit at run time:

- **0**: Fits both Axes at the same time
- **1**: Fits only the Vertical Axis
- **2**: Fits only the Horizontal Axis

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.FitPen(1)
  Chart.FitPen("Pen1", 1)
  ' Fits Pen1 only vertically
End Sub
```

4.6.2.5 LoadData

LoadData()

Loads data into an E3Chart. This method is specially used for loading data before printing, when used in a Report object.

NOTE: The **LoadData** method is synchronous only if a Pen is not in **Automatic** mode.

4.6.2.6 ResetConfig

ResetConfig([*Flags*])

Removes all settings configured in an E3Chart, rolling it back to its initial status.

NOTE: This method also has an optional parameter called *Flags*, which is kept only for compatibility reasons.

4.6.2.7 ShowCursors

ShowCursors()

Activates the **Interval Search** mode. At run time, this feature can be accessed by right-clicking the object and then selecting the **Interval Search** option on its

contextual menu.

4.6.2.8 ZoomIn

ZoomIn()

The **ZoomIn** method increases zoom in an E3Chart, that is, it closes up Pen viewing. At run time, this feature can be accessed by right-clicking the object and then selecting the **Zoom In** option on its contextual menu.

4.6.2.9 ZoomOut

ZoomOut()

The **ZoomOut** method decreases zoom in an E3Chart, that is, it opens up Pen viewing in E3Chart. At run time, this feature can be accessed by right clicking an E3Chart and then selecting the **Zoom Out** option on its contextual menu.

4.6.3 Properties

This section contains information about the properties of the E3Chart object.

4.6.3.1 Axes

Returns a collection of E3Chart's Axes. Then, all properties of this Axes collection can be modified.

4.6.3.2 BackColor

Determines a background color for an E3Chart. For this color to be displayed, the **ShowBackground** property must be configured as True. Default value of this property is beige (**RGB(236, 233, 216)**).

4.6.3.3 CursorBegin

Defines the initial cursor position, between 0 (zero) and 1 (one). Users must execute the **ShowCursors** method or enable the **Interval Search** option for these cursors to appear.

4.6.3.4 CursorColor

Establishes a color for an interval search cursor. Default value is red (**RGB(255, 0, 0)**).

4.6.3.5 CursorEnd

Defines the final cursor position, between 0 (zero) and 1 (one). Users must execute the **ShowCursors** method or enable the **Interval Search** option for these cursors to appear.

4.6.3.6 CursorLineStyle

Line style for an interval search cursor. The available options are described on the next table.

Available options for CursorLineStyle

OPTION	DESCRIPTION
0 - LS_Solid	Applies a solid line on an E3Chart's interval cursor.
1 - LS_Dash	Applies a dashed line on an E3Chart's interval cursor.
2 - LS_Dot	Applies a dotted line on an E3Chart's interval cursor.
3 - LS_Dashdot	Applies a dash-dotted line on an E3Chart's interval cursor.
4 - LS_Dashdotdot	Applies a dash-dot-dot line on an E3Chart's interval cursor.
5 - LS_Null	Applies an invisible line on an E3Chart's interval cursor.

4.6.3.7 CursorLineWidth

Establishes the width of an interval cursor.

4.6.3.8 CursorSearchStyle

Allows a cursor to search for chart points, according to the following options:

- **0 - PointNearest**: Searches for the nearest point
- **1 - LinearInterpolation**: Searches for an interpolated point
- **2 - PointPrevious**: Searches for the previous point

4.6.3.9 ForeColor

Determines a color for an E3Chart's foreground. Default value of this property is black (**RGB(0, 0, 0)**).

4.6.3.10 GridBkColor

Determines a color for an E3Chart grid's background. Default value of this property is white (**RGB(255, 255, 255)**). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.GridBkColor
  MsgBox "Next"
  E3Chart1.GridBkColor = RGB(0, 255, 0)
  MsgBox "Return"
  E3Chart1.GridBkColor = Old
End Sub
```

4.6.3.11 HorAxisTitle

Determines a title for the main Horizontal Axis. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1= Screen.Item("E3Chart1")
  Old = E3Chart1.HorAxisTitle
  MsgBox "Next"
  E3Chart1.HorAxisTitle = "!Test"
  MsgBox "Return"
  E3Chart1.HorAxisTitle = Old
End Sub
```

4.6.3.12 HorGrid

Determines the line type applied to an E3Chart's horizontal grid. The available options are described on the next table.

Available options for HorGrid

OPTION	DESCRIPTION
0 - Solid	Applies a solid line on an E3Chart's horizontal grid.
1 - Dash	Applies a dashed line on an E3Chart's horizontal grid.
2 - Dot	Applies a dotted line on an E3Chart's horizontal grid (default).
3 - Dashdot	Applies a dash-dot line on an E3Chart's horizontal grid.
4 - Dashdotdot	Applies a dash-dot-dot line on an E3Chart's horizontal grid.
5 - Invisible	Applies an invisible line on an E3Chart's horizontal grid.

Example:

```
Sub CommandButton1_Click()
```

```
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorGrid
  For i = 0 To 5
    E3Chart1.HorGrid = i
    MsgBox "E3Chart1.HorGrid =" & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.HorGrid = Old
End Sub
```

4.6.3.13 HorGridColor

Determines a color for an E3Chart's horizontal grid. Default value of this property is gray (**RGB(192, 192, 192)**). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorGridColor = RGB(255, 0, 0)
  MsgBox "Next"
  E3Chart1.HorGridColor = RGB(255, 0, 0)
  MsgBox "Next"
  E3Chart1.HorGridColor = RGB(0, 0, 255)
  MsgBox "Return"
  E3Chart1.HorGridColor = Old
End Sub
```

4.6.3.14 HorMinorTicks

Determines the number of subdivisions of horizontal scales of a grid. Default value of this property is 1 (one). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorMinorTicks
  For i = 0 To 5
    E3Chart1.HorMinorTicks = i
    MsgBox "Next value"
  Next
  E3Chart1.HorMinorTicks = Old
End Sub
```

4.6.3.15 HorScaleBegin

Determines an initial value applied on a grid's main horizontal scale. This value can either be numerical for XY E3Charts or data for E3Charts with fixed-time scales. For real-time E3Charts, this property is not applied, which uses the **TimeSpan** property instead.

4.6.3.16 HorScaleEnd

9 Determines the final value applied on a grid's main horizontal scale. This value can either be numerical for XY E3Charts or data for E3Charts with fixed-time scales. For real-time E3Charts this property is not applied, which uses the **TimeSpan** property instead. Example:

```
Sub ComboBox1_Change()
  ' Defines which query to display
  current_query_index = ListIndex
  Set E3Chart1 = Screen.Item("E3Chart1")
  i = 0
  For Each query In E3Chart1.Queries
    If i = current_query_index Then
      ' Retrieves everything
      query.FieldFilter(0) = "" 'Retrieves all
      Set current_query = query
    Else
      ' Retrieves everything
      ' to prevent slowing down the operation
      query.FieldFilter(0) = "<0" 'Do not retrieve
      ' anything, to prevent slowness
    End If
    i = i + 1
  Next
  ' Only displays pens using the current query
  For Each pen In E3Chart1.Pens
    pen.Visible = (pen.QueryName = current_query.Name)
  Next
  ' Updates all queries
  E3Chart1.Queries.UpdateData()
  Screen.Item("E3Chart1").HorScaleBegin = Now - 0.001
  Screen.Item("E3Chart1").HorScaleEnd = Now
End Sub
```

4.6.3.17 HorScaleFormat

A Contains a text that represents a mask, in which all values of an horizontal scale are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers with signal or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date and time values (when valid). To represent only time, use the equivalent format

- **Time**: Displays numerical time and date values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal for numbers with variable magnitude. Allows up to 15 decimal places
- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

A mask for these formats, as specified in the **Type** field, is displayed in the Properties Window (for example, **M/d/yy H:mm**, **0E-00**, etc.).

4.6.3.18 HorTickUnit

9 Determines the number of subdivisions between grid marks. When this property is equal to 0 (zero), spacing is automatic. Example:

```
Sub SubCommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.HorTickUnit
  For i = 0 To 30 Step 10
    E3Chart1.HorTickUnit = i
    MsgBox "E3Chart1.HorTickUnit = " & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.HorTickUnit = Old
End Sub
```

4.6.3.19 Legend

Returns an E3Chart's Legend object. Then, all Legend properties can be changed.

4.6.3.20 MouseMode

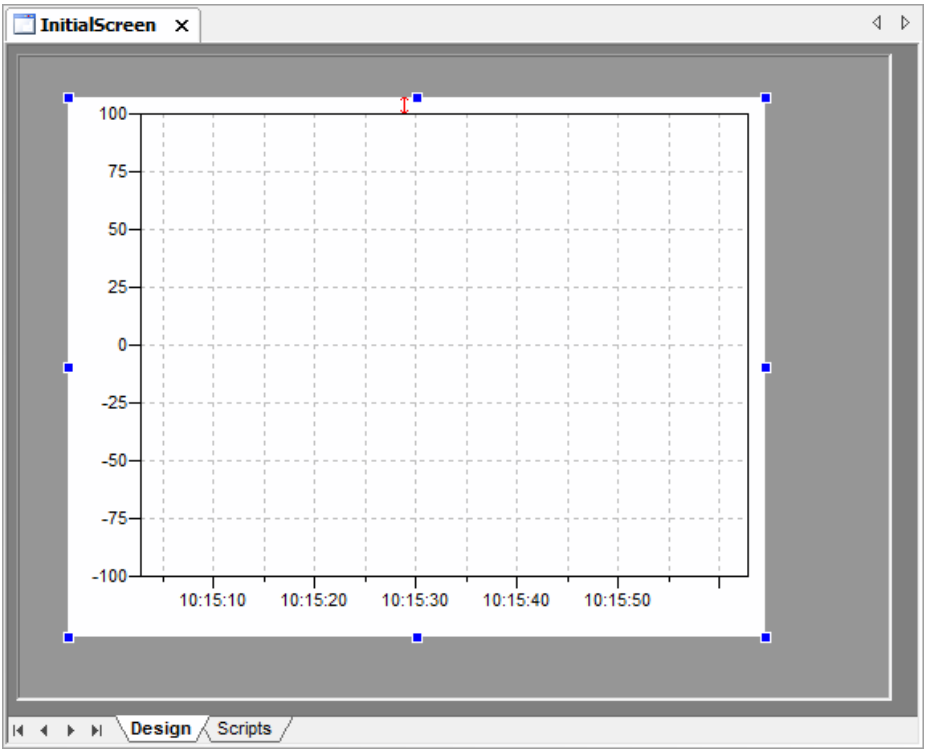
Selects one of the runtime options of an E3Chart's contextual menu. Possible values for this property are the following:

- **0 - MouseModeZoom**: Places the mouse pointer in **Zoom** mode by selected area. This option is available in numerical scale charts in XY and fixed scale. This is equivalent of selecting the **Zoom Box** option on E3Chart's runtime menu
- **1 - MouseModePan**: Places the mouse pointer in scale's movement mode. This is equivalent of selecting the **Move** option on E3Chart's runtime menu

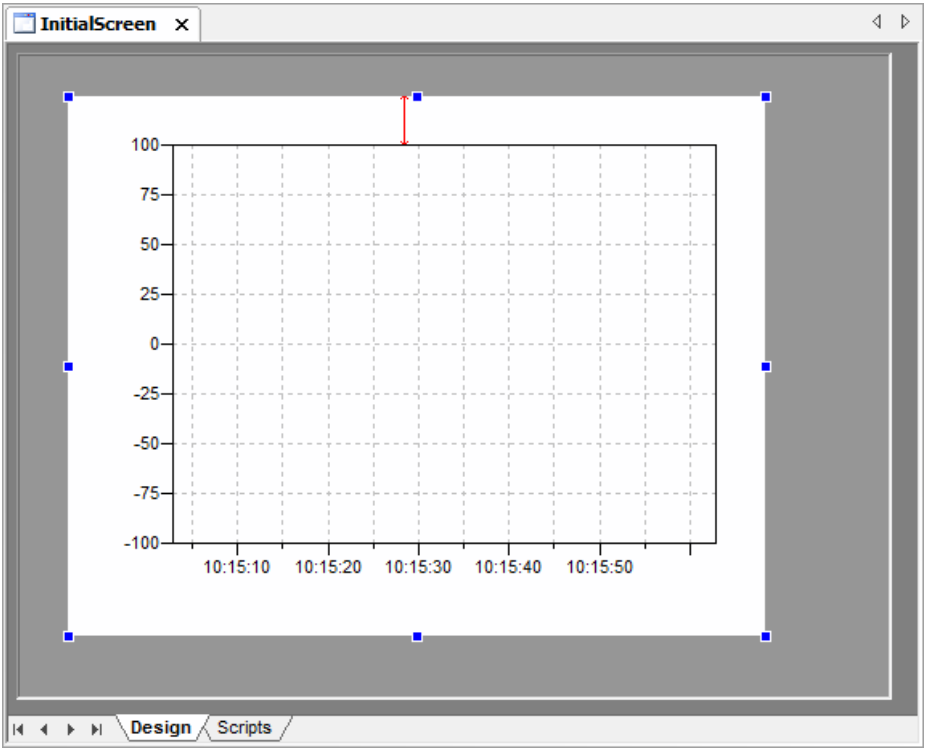
- **2 - MouseModePanH**: Places the mouse pointer in scale's movement mode only in horizontal direction. This is equivalent of selecting the **Move horizontally** option on E3Chart's runtime menu
- **3 - MouseModeSearch**: Places the mouse pointer in value search mode for Pen's data. This is equivalent of selecting the **Search** option on E3Chart's runtime menu
- **4 - MouseModeCursors**: Enables the time interval search option. This is equivalent of selecting the **Search Intervals** option on E3Chart's runtime menu

4.6.3.21 Padding

9 This property determines the distance, in pixels, between a chart and an E3Chart's border, as seen on the next figures with red arrows. Default value of this property is 10. Example:



Padding property equal to 10



Padding property equal to 30

4.6.3.22 Pens

Returns an E3Chart's Pen Collection object. A Pen Collection object is used to insert, remove, or access all available Pens of an E3Chart. This is a read-only property. Example:

```
Sub CommandButton1_Click()
  For Each pen In Screen.Item("E3Chart1").Pens
    pen.Visible = True
  Next
End Sub
```

4.6.3.23 Queries

Returns an E3Chart's Query Collection object. A Query Collection object is used to insert, remove, or access all available Queries of an E3Chart. This is a read-only property.

4.6.3.24 RefreshTime

This property determines an E3Chart's update time.

4.6.3.25 ScaleFont

A Determines a font for the text used on a grid. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Chart1").ScaleFont = "Times New Roman"
  Screen.Item("E3Chart1").ScaleFont.Size = 12
  Screen.Item("E3Chart1").ScaleFont.Italic = True
End Sub
```

4.6.3.26 ShowBackground

◩ Enables or disables viewing a chart's background. If this property is equal to True, chart's background is displayed. Otherwise, this chart has a transparent background. The color selected in the **BackColor** property does not appear if this property is equal to False (default). Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.ShowBackground = Not E3Chart1.ShowBackground
End Sub
```

4.6.3.27 ShowBottomScale

◩ If this property is set to True, the main Horizontal Axis is displayed at grid's bottom. Otherwise, it is not displayed. Default value is True. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  oldBottomScale = E3Chart1.ShowBottomScale
  MsgBox "Display axis"
  E3Chart1.ShowBottomScale = True
  MsgBox "Hide axis"
  E3Chart1.ShowBottomScale = False
  MsgBox "Return..."
  E3Chart1.ShowBottomScale = oldBottomScale
End Sub
```

4.6.3.28 ShowGridBackground

◩ Enables or disables viewing a grid's background. If this property is equal to True (default), a grid's background is displayed. Otherwise, a grid remains with a transparent background. The selected color in the **GridBkColor** property does not appear if this property is equal to False. Example:

```
Sub CommandButton1_Click()
  Set Chart1 = Screen.Item("E3Chart1")
  Chart1.ShowGridBackground = Not Chart1.ShowGridBackground
End Sub
```

4.6.3.29 ShowLeftScale

◩ If this property is set to True, the main Vertical Axis is displayed on the left side of the grid. Otherwise, it remains invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowLeftScale = Not Chart.ShowLeftScale
End Sub
```

4.6.3.30 ShowPopupMenu

◩ Enables or disables the option of displaying E3Chart's runtime menu. If its value is equal to True, this menu is displayed when users right-click E3Chart's chart. If its value is equal to False, this menu is not displayed. Default value of this property is True.

4.6.3.31 ShowRightScale

◩ If this property is set to True, the main Vertical Axis is displayed on the right size of the grid. Otherwise, it remains invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowRightScale = Not Chart.ShowRightScale
End Sub
```

4.6.3.32 ShowTitle

◩ If this property is set to True, E3Chart's main title is visible. Otherwise, it remains invisible. The **Title** property contains a title to display on an E3Chart. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  oldTitle = E3Chart1.Title
  oldShowTitle = E3Chart1.ShowTitle
  E3Chart1.Title = "Test!"
  MsgBox "Display"
  E3Chart1.ShowTitle = True
  MsgBox "Hide"
  E3Chart1.ShowTitle = False
  MsgBox "Return"
  E3Chart1.Title = oldTitle
  E3Chart1.ShowTitle = oldShowTitle
End Sub
```

4.6.3.33 ShowTopScale

◩ If this property is set to True, the main Horizontal Axis is displayed on top of the grid. Otherwise, it is not displayed. Default value is False. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.ShowTopScale = Not Chart.ShowTopScale
End Sub
```

4.6.3.34 TimeSpan

Indicates a time scale that appears on E3Chart's main Horizontal Axis, when it is configured to display a real-time scale. This property's value is always in seconds. Default value of this property is 60.

4.6.3.35 Title

Determines E3Chart's main title. For this title to appear on an E3Chart, the **ShowTitle** property must be set to True.

4.6.3.36 TitleFont

Determines E3Chart's main title font. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Title = "Test"
  E3Chart1.ShowTitle = True
  MsgBox "Change font"
  E3Chart1.TitleFont = "Times New Roman"
  MsgBox "Change size"
  E3Chart1.TitleFont.Size = 20
End Sub
```

4.6.3.37 VerAxisTitle

Determines a title for the main Vertical Axis.

4.6.3.38 VerGrid

Determines a type of line to apply on E3Chart's vertical grid. The available options are described on the next table.

Available options for VerGrid

OPTION	DESCRIPTION
0 - Solid	Applies a solid line on E3Chart's vertical grid.
1 - Dash	Applies a dashed line on E3Chart's vertical grid.
2 - Dot	Applies a dotted line on E3Chart's vertical grid (default).
3 - Dashdot	Applies a dash-dot line on E3Chart's vertical grid.

OPTION	DESCRIPTION
4 - Dashdotdot	Applies a dash-dot-dot line on E3Chart's vertical grid.
5 - Invisible	Applies an invisible line on E3Chart's vertical grid.

4.6.3.39 VerGridColor

Determines a color for the line on grid's main Vertical Axis. Default value of this property is gray (**RGB(192, 192, 192)**).

4.6.3.40 VerMinorTicks

Determines the number of subdivisions between marks of grid's main Vertical Axis. Default value of this property is 1 (one).

4.6.3.41 VerScaleBegin

Determines the value on top of grid's main Vertical Axis. Default value of this property is 100.

4.6.3.42 VerScaleEnd

Determines the value at the bottom of grid's main Vertical Axis. Default value of this property is -100.

4.6.3.43 VerScaleFormat

Contains a text that represents a mask, in which all values of a vertical scale are displayed. This mask can represent several types of values:

- **General**: It has no specific format, automatically adapting itself to the specified value
- **Number**: Displays numbers with an integer and a fraction part. Users may opt for up to 15 decimal places, for using a thousand separator or not, and for displaying negative numbers with signal or between parentheses. For very large or very small numbers, the **Scientific** format is recommended
- **Date**: Displays numerical date and time values (when valid). To represent only time, use the equivalent format
- **Time**: Displays numerical time and date values (when valid). To represent only dates, use the equivalent format
- **Percentage**: Multiplies the number by 100 and adds a percentage symbol. Allows up to 15 decimal places
- **Scientific**: Displays the number with a mantissa and exponent notation. Ideal

for numbers with variable magnitude. Allows up to 15 decimal places

- **Special**: Allows formatting integers on non-decimal basis (hexadecimal, octal, or binary, for example)
- **Other**: Allows directly editing the format code, or selecting a previously created format

A mask for these formats, as specified in the **Type** field, is displayed in the Properties Window (for example, **M/d/yy H:mm**, **0E-00**, etc.).

4.6.3.44 VerTickUnit

9 Determines the number of subdivisions between grid marks. When this property is set to 0 (zero), spacing is automatic. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  Old = E3Chart1.VerTickUnit
  For i = 0 To 30 Step 10
    E3Chart1.VerTickUnit = i
    MsgBox "E3Chart1.VerTickUnit = " & CStr(i)
  Next
  MsgBox "Return"
  E3Chart1.VerTickUnit = Old
End Sub
```

4.6.4 Pen Collection

This section contains information about **methods** and **properties** of the Pen Collection object. This object does not have events associated to it.

4.6.4.1 Methods

This section contains information about the methods of the Pen Collection object.

NOTE: E3Chart's Pen Collection can be accessed via **Pens** property.

4.6.4.1.1 AddPen

AddPen(*PenName*)

The **AddPen** method adds a new Pen to an E3Chart, and returns that Pen. Example:

```
Sub CommandButton1_Click()
  ' Creates an untitled Pen.
  Set Pen = Screen.Item("E3Chart1").Pens.AddPen("")
  MsgBox Pen.Name
End Sub
```

```
SubCommandButton1_DbClick()
  ' Creates a Pen named "Pen1".
  ' If this name already exists, increments it.
  Set Pen = Screen.Item("E3Chart1").Pens.AddPen("Pen1")
  MsgBox Pen.Name
End Sub

Sub CommandButton2_Click()
  ' Creates a Pen and links it to DemoTag1.
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("")
  MsgBox Pen.Name
  Pen.UsetimeStamp = True
  Pen.YLink = "Data.DemoTag1"
  Pen.Connect()
End Sub
```

4.6.4.1.2 ChangePenPos

ChangePenPos(*Source*, *Dest*)

Modify the drawing order of Pens on an E3Chart. This method has the following parameters:

- **Source**: Determines the index of a Pen to move (starting at one)
- **Dest**: Determines a Pen's destination (starting at one)

A situation in which this method is especially useful is when users have a line-shaped Pen and an area-shaped Pen. If the area-shaped Pen is drawn after the line-shaped Pen, that first Pen may hide the last one. A solution could be invert the drawing order of these Pens. Example:

```
Sub CommandButton1_Click()
  ' Moves Pen 1 to position 2.
  Screen.Item("E3Chart1").Pens.ChangePenPos(1, 2)
End Sub
```

4.6.4.1.3 Item

Item(*Index*)

The **Item** method returns a Pen object from a Pen Collection, specified by an index. This method has the *Index* parameter, which can be a number (if it corresponds to a Pen's index) or a text (if it corresponds to a Pen's name) type. Example:

```
Sub CommandButton1_Click()
  ' Gets the first Pen.
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item(0)
End Sub
```

4.6.4.1.4 Remove

Remove(*Index*)

Removes a Pen from a Pen Collection, specified by name or by index. This method has the *Index* parameter, which can be a number (if it corresponds to a Pen's index) or a text (if it corresponds to a Pen's name).

4.6.4.1.5 SetCursorPos

SetCursorPos(*X*, *Range*)

Places the cursor of every Pen on an E3Chart. This is the equivalent of using the **SetCursorPos** method for each Pen.

4.6.4.2 Properties

This section contains information about the properties of the Pen Collection object.

4.6.4.2.1 Count

9 Contains the total amount of Pens inserted on an E3Chart. This is a read-only property.

4.6.4.3 Pens

This section contains information about **methods** and **properties** of the Pen object. This object does not have events associated to it.

4.6.4.3.1 Methods

This section contains information about the methods of the Pen object.

4.6.4.3.1.1 AddPoint

AddPoint(*ValueX*, *ValueY*[, *Quality*])

Adds a point at the end of the real-time buffer. This buffer's size is only valid after connecting a Pen. If a Pen is created in Studio, its connection is automatic, but if it is created via script, it is necessary to call the **Connect** method after creating it. This method must be used with real-time Pens, and with the **UseTimeStamp** property set to False. The optional *Quality* parameter indicates the quality of a point to insert. If this parameter is not informed, this point's quality is considered as good (192). The number of points that can be added to a Pen depends on the buffer size (the Pen's **BufferSize** property).

4.6.4.3.1.2 Clear

Clear()

Erases data from real-time buffers, without decreasing its size. This method does not disconnect Links, nor removes historical data.

4.6.4.3.1.3 Connect

Connect()

The **Connect** method connects a Pen to a server to receive real-time data, linking the **XLink** and **YLink** properties. If a Pen is already connected, this method remains inactive. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Disconnect()
  Pen1.Connect()
End Sub
```

4.6.4.3.1.4 Disconnect

Disconnect()

The **Disconnect** method clears current data and prevents a Pen from receiving more real-time data from its linked Tag. If a Pen is already disconnected, this method remains inactive. When the **Disconnect** method is used with a Mixed Pen (the **DataSourceType** equal to 2), it removes its real-time part, and keeps the historical part. At run time, the **Connect** method must be used to show real-time data again. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Disconnect()
  Pen1.Connect()
End Sub
```

4.6.4.3.1.5 GetCursorPos

GetCursorPos(*X*, *Y*)

Returns the position where a cursor intercepted E3Chart's Pen. This method has the *X* and *Y* parameters, which correspond to the x and y cursor coordinates. If this method is successful it returns True, otherwise it returns False. Example:

```
Sub CommandButton1_Click()
  For Each pen In Chart.Pens
    If pen.GetCursorPos(aa, bb) Then
      strResult = strResult & pen.name & _
        " := " & CSTr(CDate(aa)) + _
        "y " + CStr(bb) + vbNewLine
```

```
    End If
  Next
  MsgBox strResult
End Sub
```

4.6.4.3.1.6 GetPoint

GetPoint(*ValueX*, *ValueY*)

Returns the X and Y coordinates of the nearest point of the input value in *ValueX*. The *ValueX* parameter informs a reference value to search for this point, and then receives the effective value of the X coordinate of the nearest point found. The *ValueY* parameter returns the effective value of the Y coordinate of the nearest point found. This method returns True if it finds a point, and False otherwise.

4.6.4.3.1.7 SetCursorPos

SetCursorPos(*X*, *Range*)

Places an E3Chart Pen's cursor. The *X* parameter indicates in which position this cursor must be placed, similar to the behavior of moving a search cursor with the mouse pointer. A cursor is moved to the nearest position indicated by *X*. The *Range* parameter is optional and used as a validation. A cursor is only moved if a valid point is inside that interval. Any negative value indicates that this interval must not be used. Example:

```
' If there is a valid point in (x = 1, y = 10)
' and another one in (x = 4, y = 20)
SetCursorPos(2) ' moves the cursor to the point (1, 10)
SetCursorPos(4) ' moves the cursor to the point (4, 20)

' When range is used, the cursor is only moved
' if the point is inside the range.
' Does not move the cursor, because 2 is more than 0.5 units
' away from the nearest point, which is 1
SetCursorPos (2, 0.5)
' Sends the cursor to the point (1, 10)
SetCursorPos (2, 4)
```

This method returns True if a cursor was moved, otherwise it returns False.

4.6.4.3.2 Properties

This section contains information about the properties of the Pen object.

4.6.4.3.2.1 AutoQuery

When the **AutoQuery** property is set to True, this is called an **AutoQuery**-type Pen, or simply an **Automatic Pen**. The goal of an Automatic Pen is to save memory

consumption and query time. To do so, it applies filters in the **E3Timestamp** field to bring only needed data into E3Chart's area. Every time an E3Chart's visible period changes, an Automatic Pen retrieves missing data to complete the drawing of that period. In addition, an Automatic Query also completes historical missing data to connect Pen's historical and real time parts. If, by any reason, historical data is not retrieved after thirty seconds, an Automatic Query is canceled on that part. Please check also the **MaxGapTime** property for more details.

Due to the way an Automatic Query applies filters to an **E3TimeStamp** field, it is not available for Storage queries or user-defined SQL code. That is, even when **AutoQuery** property is set to True, it has no effect on **Storage**-type Queries. One way to recognize an Automatic Query is a hatched drawing on the E3Chart screen. Every time hatches appear on that drawing, this means that on this period an Automatic Query is being performed. When hatch borders are in red, this means that Automatic Query on that part is on failure. In this case, E3Chart re-executes problematic Queries on that part.

NOTE: Unlike a real time part, where each Pen has its own data buffer, Pen's historical part is stored on the Query, and it is shared among Pens. For example, when a Query has three fields, **E3Timestamp**, **Field1**, and **Field2**, this data is stored on the Query and it is available for Pens sharing that Query. Thus, the common part, usually the **E3Timestamp** field, can be used by two different Pens, without duplicating data. For Automatic Pens, two different Pens can use the same table and, due to different scales, they can load different Query periods. In this situation, each Pen automatically inherits the part loaded by the other Pen.
The Query object cannot work simultaneously in **Automatic** and **Non-Automatic** mode. This also means that if different Pens, one Automatic and another one Non-Automatic, want to share the same Query, that Query is going to adapt to the first Pen using it. That is, the **AutoQuery** property does not ensure that a Query is in **Automatic** mode, this depends on other factors.

4.6.4.3.2.2 AverageY

Informs the average of a Pen in an interval, if the **EnableCalc** property is enabled. If an E3Chart is in **Interval Search** mode, displays the average of this interval. Otherwise, displays the interval between the beginning and the end on the Horizontal Axis. Values with bad quality are not considered if the **ShowBadPoints** property is disabled. This is a read-only property.

4.6.4.3.2.3 BkColor

Determines a background color used on an **Area**-type Pen. Default value of this property is empty.

4.6.4.3.2.4 BufferSize

Determines the number of points kept on a real-time Pen. After this value, older data is discarded. On historical Pens, this property has no effect. This property is only considered after connecting a Pen. For more information, please check the **Connect** method. Default value of this property is 1000, and it must be always greater than 0 (zero). Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Disconnect()
  Pen1.BufferSize = 5000
  Pen1.Connect()
End Sub
```

4.6.4.3.2.5 Color

Determines a color for a Pen's line on an E3Chart. Default value of this property is empty.

4.6.4.3.2.6 DataSourceType

Determines the source of Pen's data. The available options for this property are described on the next table.

Available options for DataSourceType property

OPTION	DESCRIPTION
0 - Real Time	Indicates connection of this Pen to a Tag updated in real-time.
1 - Historic	Indicates connection of this Pen to data from a query.
2 - Real Time & Historic	Indicates connection of this Pen to real-time Tags and historical data simultaneously.

When the **DataSourceType** property is equal to 0 (zero, Real Time), the **XLink** and **YLink** properties inform all links used, or else the **UseTimeStamp** property informs that the **XLink** property is not used, being replaced by the **YLink** property's timestamp. When the **DataSourceType** property is equal to 1 (one, Historical), the **XField** and **YField** properties inform field tables to use. The **QueryName** property indicates the name of the table used. When the **DataSourceType** property is equal to 2 (two, Real Time & Historical), the 0 (zero) and 1 (one) options work simultaneously for this Pen.

NOTE: At run time, when this property changes and a Pen stops displaying real-time data, users must call the **Connect** method to display that data again.

Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  MsgBox "Click OK to create a Pen."
  Set Pen = E3Chart1.Pens.AddPen("Pen1")
  Pen.DataSourceType = 0 ' Real time
  Pen.YLink = "Data.DemoTag1"
  Pen.UseTimeStamp = True ' In X uses timestamp
  Pen.Color = RGB(255, 0, 0)
  Pen.Docstring = "Test"
  MsgBox "Click OK to connect."
  Pen.Connect() ' Starts receiving data
  MsgBox "Click OK to fit."
  E3Chart1.FitPen(0)
  MsgBox "Click OK to remove this pen."
  E3Chart1.Pens.Remove(Pen.Name)
End Sub
```

4.6.4.3.2.7 DigitalData

◩ Determines a digital plot style. If this property is set to True, the digital plot style assumes that data variation is in digital form, that is, its value relative to the last one varied instantly. Otherwise, variation is considered linear and points are connected by a line segment. Default value of this property is True.

4.6.4.3.2.8 EnableCalc

◩ Enables or disables the calculation of average, minimum, and maximum inside an interval.

4.6.4.3.2.9 EnableHighLimit

◩ Enables or disables a check on the high limit.

4.6.4.3.2.10 EnableLowLimit

◩ Enables or disables a check on the low limit.

4.6.4.3.2.11 EU

A This property is used to identify the engineering unit that this value represents, such as degrees, meters, KW/h, etc.

4.6.4.3.2.12 HighlightMaxGapTime

Specifies if a line that visually connects the historical part and the real-time part of a Mixed Pen (defined in **MaxGapTime**) must have a different color (defined in **MaxGapTimeColor**) and style (defined in **MaxGapTimeStyle**). Default value of this property is False.

4.6.4.3.2.13 HighLimit

Determines the limit of a high alarm.

4.6.4.3.2.14 InterpolatedBeginY

Informs the value of an interpolated point where the initial cursor crosses a Pen. This is a read-only property.

4.6.4.3.2.15 InterpolatedEndY

Informs the value of an interpolated point where the final cursor crosses a Pen. This is a read-only property.

4.6.4.3.2.16 LimitPenBkColor

Determines a Pen's background color when in alarm.

4.6.4.3.2.17 LimitPenColor

Determines a Pen's color when in alarm.

4.6.4.3.2.18 LowLimit

Determines the limit of a low alarm.

4.6.4.3.2.19 MaxGapTime

Allows specifying a limit time to consider when visually connecting the historical part and the real-time part of a Mixed Pen. Default value of this property is 0 (zero). This property's value can be modified at run time.

NOTE: This property is available in version 3.5 or later. For applications created in earlier versions and opened in version 3.5 or later, this property's value is always 0 (zero).

4.6.4.3.2.20 MaxGapTimeColor

Allows configuring a color for a visual connection between the historical part and the real-time part of a Mixed Pen, defined in the **MaxGapTime** property. Default value of this property is red (**RGB(255, 0, 0)**).

4.6.4.3.2.21 MaxGapTimeStyle

Specifies a line style that establishes a visual connection between the historical part and the real-time part of a Mixed Pen, configured in the **MaxGapTime** property. Possible values for this property are the following:

- **0**: Solid
- **1**: Dashed
- **2**: Dotted
- **3**: Dash - Dot
- **4**: Dash - Dot - Dot
- **5**: Invisible

NOTE: The **5** (Invisible) option of this property can only be selected via script.

4.6.4.3.2.22 MaxY

Informs a Pen's maximum value in an interval, if the **EnableCalc** property is enabled. If an E3Chart is in **Interval Search** mode, displays the average in this interval. Otherwise, displays the average in the interval between the beginning and the end in the Horizontal Axis. Values with bad quality are not considered if the **ShowBadPoints** property is disabled. This is a read-only property.

4.6.4.3.2.23 MinY

Informs a Pen's minimum value in an interval, if the **EnableCalc** property is enabled. If an E3Chart is in **Interval Search** mode, displays the average in this interval. Otherwise, displays the average in the interval between the beginning and the end in the Horizontal Axis. Values with bad quality are not considered if the **ShowBadPoints** property is disabled. This is a read-only property.

4.6.4.3.2.24 Name

Determines a Pen's name.

4.6.4.3.2.25 PenStyle

Determines a Pen's line style. Default value of this property is 0 (zero). The available options are described on the next table.

Available options for PenStyle property

OPTION	DESCRIPTION
0 - LsSolid	Solid line
1 - LsDash	Dashed line
2 - LsDot	Dotted line
3 - LsDashDot	Dash-dot line
4 - LsDashDotDot	Dash-dot-dot line
5 - LsNull	No line

NOTE: Using a value different from 0 (zero, **LsSolid**) in this property, combined with the **Width** property with values greater than 1 (one), may degrade Pen's drawing performance.

4.6.4.3.2.26 PenType

Determines the drawing type of a Pen on an E3Chart:

- **0**: Line
- **1**: Point
- **2**: Point and Line
- **3**: Area

4.6.4.3.2.27 QueryName

Determines the name of a Query used by a Pen. This property is used if the **DataSourceType** property is set to 1 (one, Historic).

4.6.4.3.2.28 ScaleX

The **ScaleX** property indicates to which E3Chart's X scale this Pen is linked. The scale configured for **ScaleX** has a horizontal orientation, that is, it may be positioned on top or at the bottom of an E3Chart. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("DemoTagPen2")
  Pen.XLink = "Data.DemoTag2"
  Pen.UseTimeStamp = True
  ' The scale must exist.
  Pen.ScaleX = "ScaleForDemoTag2"
```

```
  Pen.Connect
End Sub
```

4.6.4.3.2.29 ScaleY

9 The **ScaleY** property indicates to which E3Chart's Y scale this Pen is linked. The scale configured for **ScaleY** has a vertical orientation, positioned on the left or on the right of an E3Chart. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set Pen = Chart.Pens.AddPen("DemoTagPen2")
  Pen.YLink = "Data.DemoTag2"
  Pen.UseTimeStamp = True
  ' The scale must exist.
  Pen.ScaleY = "ScaleForDemoTag2"
  Pen.Connect
End Sub
```

4.6.4.3.2.30 ScanValue

9 Defines an expected reading time for a real-time Pen's Tag. This value is considered for an analog-drawing mode. When this value overrides the value determined by **ScanValue**, it is considered that a Tag's value was not changed on that interval. Otherwise, when **ScanValue** is equal to 0 (zero), Pen data is always connected with a straight line joining these two points, as if that value varies linearly. This property's unit is in milliseconds.

4.6.4.3.2.31 ShowAverage

◩ Enables the display of Pen's average on an E3Chart. This property is only effective if the **EnableCalc** property is enabled.

4.6.4.3.2.32 ShowBadPoints

◩ When disabled, bad quality points are not plotted. If a Pen is drawing lines, the lines that cross bad quality points are not connected. For a point's quality to be considered in a Pen's historical part, the **fieldname_quality** field must be selected on E3Chart's Query. When enabled, all points are plotted normally.

4.6.4.3.2.33 ShowMinMax

◩ Enables the display of minimum and maximum points of an E3Chart's Pen. This property is only effective if the **EnableCalc** property is enabled.

4.6.4.3.2.34 UseTimeStamp

Determines that, for an Horizontal Axis, a timestamp value linked to an Vertical Axis must be used. Please check an example in the description of the

DataSourceType property.

4.6.4.3.2.35 Visible

◩ Determines if a Pen is visible on an E3Chart. If this option is set to True, a Pen is visible at run time. Otherwise, a Pen remains invisible. Example:

```
Sub CommandButton1_Click()
  Set Pen1 = Screen.Item("E3Chart1").Pens.Item("Pen1")
  Pen1.Visible = Not Pen1.Visible
End Sub
```

4.6.4.3.2.36 Width

9 Determines the width of a Pen's line on an E3Chart. Default value of this property is 0 (zero).

NOTE: Using values greater than 1 (one) in this property, combined with the **PenStyle** property with a value different from 0 (zero, **LsSolid**), may degrade Pen's drawing performance.

4.6.4.3.2.37 XField

A Name of a Query Field used to plot data on a horizontal scale. It is used for historical Pens.

4.6.4.3.2.38 XLink

A Name of a link used to plot data on a horizontal scale. When this property's value changes, a Pen is automatically disconnected. After configuring it, users must call the **Connect** method so that this Pen starts receiving data relative to this link. This property is used for real-time Pens.

4.6.4.3.2.39 XMaxY

9 Informs the X value relative to the **MaxY** point. This is a read-only property.

4.6.4.3.2.40 XMinY

9 Informs the X value relative to the **MinY** point. This is a read-only property.

4.6.4.3.2.41 YField

A Name of a Query field used to plot data on a vertical scale. It is used for historical Pens.

4.6.4.3.2.42 YLink

A Name of a link used to plot data on a vertical scale. When this property's value changes, a Pen is automatically disconnected. After configuring it, users must call the **Connect** method so that this Pen starts receiving data relative to this link. This is used for real-time Pens.

4.6.5 Axis Collection

This section contains information about **methods** and **properties** of the Axis Collection object. This object does not have events associated to it.

4.6.5.1 Methods

This section contains information about the methods of the Axis Collection object.

4.6.5.1.1 AddAxis

AddAxis(*AxisName*)

Adds a new Axis with the name specified in the *AxisName* parameter and returns that Axis. If users try to create an Axis with an already existing name, an error message is displayed. To generate a name automatically, users must pass a blank name in the *AxisName* parameter. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  newAxis.Color = RGB(255, 0, 0)
End Sub
```

4.6.5.1.2 Remove

Remove(*Index*)

Removes an Axis by its name or index, as specified in the *Index* parameter. Main Axes 0 (zero) and 1 (one) cannot be removed. If users try to remove them, an error message is displayed. Example:

```
Sub CommandButton1_Click()
  ' This example removes all additional axes
  Set Chart = Screen.Item("E3Chart")
  While (Chart.Axes.Count > 2)
    Chart.Axes.Remove(2)
  Wend
End Sub

Sub CommandButton1_Click()
  ' Removes an additional axis, if available
  Set Chart = Screen.Item("E3Chart1")
```

```
  Chart.Axes.Remove(2)
End Sub
```

4.6.5.2 Properties

This section contains information about the properties of the Axis Collection object.

4.6.5.2.1 Count

Returns the total amount of Axes on an E3Chart, including the main Axes (Horizontal and Vertical).

4.6.5.2.2 HorAxis

Returns the main Horizontal Axis. This Axis also appears on the Axes list. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.HorAxis.Name & ", " & _
    Chart.axes.Item("AxisName").Name
End Sub
```

4.6.5.2.3 Item

Returns an Axis using its name or index. The index 0 (zero) is always the main Horizontal Axis and the index 1 (one) is always the main Vertical Axis. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart")
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.HorAxis.Name & ", " & _
    Chart.axes.Item("AxisName").Name
End Sub
```

4.6.5.2.4 VerAxis

Returns the main Vertical Axis. This Axis also appears on the Axes list. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  MsgBox Chart.axes.Item(0).Name & ", " & _
    Chart.axes.Item(1).Name
  MsgBox Chart.axes.Item("AxisName").Name & ", " & _
    Chart.axes.VerAxis.Name
End Sub
```

4.6.5.3 Axes

This section contains information about **methods** and **properties** of the Axis object. This object does not have events associated to it.

4.6.5.3.1 Methods

This section contains information about the methods of the Axis object.

NOTE: **HorAxis** and **VerAxis** are properties from the Axes Collection that access default Vertical and Horizontal Axes, respectively. For example, instead of using "Chart.Axes.Item('HorizontalAxis')", users can use "Chart.Axes.HorAxis". Other user-created Axes have their own names.

4.6.5.3.1.1 GetHistoricPeriod

GetHistoricPeriod(*Begin*, *End*)

Returns a time interval displayed on a historic scale. The *Begin* parameter indicates the historic scale's initial date and the *End* parameter indicates the final date. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart")
  Chart.Axes.Item("AxisName").GetHistoricPeriod min, max
  Value = CStr(dmin) & " " & CStr(dmax)
  MsgBox "Start Date = " & CStr(min) & _
    vbNewLine & " Ending Date = " & CStr(max)
End Sub
```

4.6.5.3.1.2 GetMinMax

GetMinMax(*Min*, *Max*)

Returns the minimum and maximum values of a numeric scale in the *Min* and *Max* parameters, respectively. Example:

```
Sub CommandButton1_DBClick()
  Set Chart = Screen.Item("E3Chart")
  Chart.Axes.Item("AxisName").GetMinMax dmin, dmax
  MsgBox CStr(dmin) & " " & CStr(max)
End Sub
```

4.6.5.3.1.3 GetRealTimePeriod

GetRealTimePeriod(*Period*)

Returns a time unit configured on a real-time scale. The *Period* parameter receives the value of a time scale. Time units available are described on the next table.

Available time units

VALUE	DESCRIPTION
0 - tuSeconds	Time unit in seconds
1 - tuMinutes	Time unit in minutes
2 - tuHours	Time unit in hours
3 - tuDays	Time unit in days
4 - tuWeeks	Time unit in weeks
5 - tuMonths	Time unit in months
6 - tuYears	Time unit in years

Example:

```
Dim Unit, Value
Unit = Screen.Item("E3Chart1").Axes.Item_
  ("HorizontalAxis").GetRealTimePeriod(Value)
MsgBox "Value: " & CStr(Value) & " Unit: " & CStr(Unit)
```

4.6.5.3.1.4 GetTickSpacing

GetTickSpacing(*TickSpacing*, *TimeUnit*)

Returns the spacing between ticks (scale subdivisions) and a configured unit. The *TickSpacing* parameter determines the spacing between ticks and the *TimeUnit* parameter determines its unit. When this parameter is zero, this means that it is automatic. This unit is not used when scale is numeric. The available values for the *TimeUnit* parameter are described on the **GetRealTimePeriod** method. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Unitvalue_
    = Chart.Axes.Item("AxisName").GetTickSpacing(TickSpacing)
  MsgBox "Value " = " & CStr(TickSpacing) & _
  " unit " & CStr(Unitvalue)
End Sub
```

4.6.5.3.1.5 SetHistoricPeriod

SetHistoricPeriod(*Begin*, *End*)

Configures a time period for a historic scale. The *Begin* parameter determines the scale's initial period and the *End* parameter determines the scale's final period. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart2")
  Chart.Axes.Item("AxisName").ScaleType = 2
  ' Displays the last period of time
  Chart.Axes.Item("AxisName").SetHistoricPeriod now - 1, now
End Sub
```

4.6.5.3.1.6 SetMinMax

SetMinMax(*Min*, *Max*)

Configures the minimum and maximum values for a numeric scale. The minimum value is determined by the *Min* parameter and the maximum value is determined by the *Max* parameter. Example:

```
Sub Circle1_Click()
  Set Chart = Screen.Item("E3Chart2")
  Chart.Axes.Item("AxisName").SetMinMax -10, 500
End Sub
```

4.6.5.3.1.7 SetRealTimePeriod

SetRealTimePeriod(*Times*, *TimeUnit*)

Configures a time period in the unit defined by the *TimeUnit* parameter. The available options in this parameter are described on the **GetRealTimePeriod** method. The *Times* parameter determines a time interval and the scale unit is specified by the *TimeUnit* parameter. An Axis is always updated in this mode (real-time). Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart5")
  ' tuSeconds = 0, tuMinutes =1, tuHours = 2, tuDays = 3,
  ' tuWeeks = 4, tuMonths = 5, tuYears = 6
  ' 2 minutes
  Chart.Axes.Item("AxisName").SetRealTimePeriod 2, 1
  Chart.Axes.Item("AxisName").SetTickSpacing 30, 0
End Sub
```

4.6.5.3.1.8 SetTickSpacing

SetTickSpacing(*TickSpacing*, *TimeUnit*)

Configures the spacing between ticks (scale subdivisions) by using the configured unit. The spacing between ticks is determined by the *TickSpacing* parameter. The *TimeUnit* parameter determines its unit. If a scale is numeric, this unit is not considered. The available options in the *TimeUnit* parameter are described on the **GetRealTimePeriod** method. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  ' 10 (if scale is numeric,
  ' the value of the unit is not considered)
  Chart.Axes.Item("AxisName").SetTickSpacing 10, 0
  Chart.Axes.Item("AxisName").SetTickSpacing 20, 0
End Sub
```

4.6.5.3.2 Properties

This section contains information about the methods of the Axis object.

NOTE: **HorAxis** and **VerAxis** are properties from the Axes Collection that access default Vertical and Horizontal Axes, respectively. For example, instead of using "Chart.Axes.Item('HorizontalAxis')", users can use "Chart.Axes.HorAxis". Other user-created Axes have their own names.

4.6.5.3.2.1 Color

Determines an Axis' main color.

4.6.5.3.2.2 EnableTextColor

This property, when enabled, specifies that an Axis' text has the same color of the scale configured in the **Color** property. Default value of this property is False.

4.6.5.3.2.3 Format

Determines a format for Axis' values. Formats allowed in this property are described on **E3 User's Manual**, on topic **Screens and Screen Objects - Value Formatting**, or leave it blank to select the **Automatic** mode. This property allows using the | (vertical bar) character on the formatting **String** as a line break.

If Axis formatting is selected as **Automatic** (Axis Properties window, **Scale** tab, **Formatting** group) and the scale type is selected as **Show latest data (Real-Time)** or **Time interval (historical data)**, date and time format obeys regional configuration of the Windows user. If the scale type is selected as numeric Scale, then it uses an automatic number format. Example:

```
Sub CommandButton1_Click()
  ' Changes formatting
  Set Chart = Screen.Item("E3Chart1")
  strOldFormat = Chart.Axes.Item("AxisName").Format
  MsgBox "Click to set automatic formatting."
  Chart.Axes.Item("AxisName").Format = "" ' Automatic
  MsgBox "Click to use another formatting."
  Chart.Axes.Item("AxisName").Format = "0.0"
  MsgBox "Click to use another formatting."
  Chart.Axes.Item("AxisName").Format = "dd/MM/yy hh:mm:ss"
  MsgBox "Click again to change back
    to the original formatting."
  Chart.Axes.Item("AxisName").Format = strOldFormat
End Sub
```

4.6.5.3.2.4 GridColor

Determines a color for grid lines. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").ShowGrid = False
  MsgBox "Click to change the color of grid lines."
  Chart.Axes.Item("AxisName").GridColor = RGB(0, 0, 255)
  Chart.Axes.Item("AxisName").ShowGrid = True
End Sub
```

4.6.5.3.2.5 GridStyle

Determines a style for grid line. The available options are described on the next table.

Available options for GridStyle

OPTION	DESCRIPTION
0 - solid	Style for grid lines is solid
1 - dash	Style for grid lines is dashed
2 - dot	Style for grid lines is dotted
3 - dashdot	Style for grid lines is dash-dot
4 - dashdotdot	Style for grid lines is dash-dot-dot
5 - invisible	There is no visible lines on the grid (invisible)

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  ' Solid 0, dash 1, dot 2, dash-dot 3,
  ' dash-dot-dot 4, invisible 5
  For i = 0 To 5
    MsgBox "Click to change the line style of the grid."
    Chart.Axes.Item("AxisName").GridStyle = i
  Next
End Sub
```

4.6.5.3.2.6 Inverse

Inverts the order of the minimum and maximum values on a numerical scale. Normally, in vertical scales, the minimum value appears at the bottom and the maximum value at the top. On horizontal scales, the minimum value appears on the left and the maximum value on the right. When the **Inverse** property is set to True, however, this order is inverted, maximum values at the bottom or on the left and minimum values at the top or on the right. Example:

```
Sub CommandButton1_Click()
```

```
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Inverse = Not _
    Chart.Axes.Item("AxisName").Inverse
End Sub
```

4.6.5.3.2.7 MinorTicks

9 Determines the total amount of subdivisions among scales. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").MinorTicks = _
    Chart.Axes.Item("AxisName").MinorTicks + 1
End Sub
```

4.6.5.3.2.8 Mirror

◩ Indicates the Axis mirroring. If this property is set to True, an Axis is mirrored on the opposite side of the original Axis. Otherwise, the Axis remains on the same position. Example:

```
Sub CommandButton1_DBClick()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Mirror = Not _
    Chart.Axes.Item("AxisName").Mirror
End Sub
```

4.6.5.3.2.9 Name

A Determines an Axis' name.

4.6.5.3.2.10 Position

Determines the position of an Axis relative to E3Chart's grid. The available options are described on the next table.

Available options for Position

OPTION	DESCRIPTION
0 - axpLeft	Axis is placed on scale's left side
1 - axpRight	Axis is placed on scale's right side
2 - axpTop	Axis is placed at scale's top
3 - axpBottom	Axis is placed at scale's bottom

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  For i = 0 To 3
    MsgBox "Click OK to change axis position."
```

```
    newAxis.Position = i
  Next
  MsgBox "Remove an axis."
  Chart.Axes.Remove(newAxis.Name)
End Sub
```

4.6.5.3.2.11 ScaleType

Determines a type of scale to display by an Axis. The available options for this item are described on the next table.

Available options for ScaleType

OPTION	DESCRIÇÃO
0 - atNumberScale	Numerical scale
1 - atLastPeriod	Displays the last period (Real Time)
2 - atPeriod	Time interval (Historic)

Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Set newAxis = Chart.Axes.AddAxis("")
  For i = 0 To 2
    MsgBox "Click OK to change scale type."
    newAxis.ScaleType = i
  Next
  MsgBox "Remove an axis."
  Chart.Axes.Remove(newAxis.Name)
End Sub
```

4.6.5.3.2.12 ShowGrid

Determines if grid lines are visible. If this property is set to True, grid lines are displayed. Otherwise, grid lines are hidden. Example:

```
Sub CommandButton_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").ShowGrid = Not _
  Chart.Axes.Item("AxisName").ShowGrid
End Sub
```

4.6.5.3.2.13 Title

Determines an Axis' title. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item("E3Chart1")
  Chart.Axes.Item("AxisName").Title = _
    Chart.Axes.Item("AxisName").Name
  MsgBox "Click to remove the title."
  ' Removes the title
```

```
  Chart.Axes.Item("AxisName").Title = ""
End Sub
```

4.6.5.3.2.14 Visible

Determines Axis' visibility on a grid. If this property is set to True, this Axis is visible on a grid. Otherwise, this Axis remains invisible. Example:

```
Sub CommandButton1_Click()
  Set Chart = Screen.Item(E3Chart1)
  Chart.Axes.Item("AxisName").Visible = Not _
  Chart.Axes.Item("AxisName").Visible
End Sub
```

4.6.6 Query Collection

This section contains information about **methods** and **properties** of the Query Collection object. This object does not have events associated to it.

4.6.6.1 Methods

This section contains information about the methods of the Query Collection object.

4.6.6.1.1 AddQuery

AddQuery(*QueryName*[, *IsInternal*])

Adds a Query to E3Chart's Query Collection. This method has the *QueryName* parameter, which determines the name of a Query being added, and *IsInternal*, which is optional, deprecated, and must not be informed.

4.6.6.1.2 Item

Item(*Index*)

The Item method returns a Query object from Query Collection, specified by index. This method has the *Index* parameter, which can be a number (if it corresponds to Query's index) or text (if it corresponds to Query's name) type.

4.6.6.1.3 Remove

Remove(*Index*)

Removes a Query object specified by name or by index from Query Collection. This method has the *Index* parameter, which can be a number (if it corresponds to Query's index) or text (if it corresponds to Query's name) type.

4.6.6.1.4 UpdateData

UpdateData()

Updates data in all Queries. Example:

```
Sub Text1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Queries.UpdateData()
End Sub
```

4.6.6.2 Properties

This section contains information about the properties of the Query Collection object.

4.6.6.2.1 Count

9 Contains the total amount of E3Chart Queries. This is a read-only property.

4.6.7 Legend

This section contains information about **methods** and **properties** of the Legend object. This object does not have events associated to it.

4.6.7.1 Methods

On a Legend, several columns can be selected. Each column displays a type of information and has a corresponding name and value. The following table displays all possible columns on a Legend.

Available options to identify a Legend column

OPTION	VALUE	NAME	DESCRIPTION
AverageY	**10**	**AverageY**	Shows Pen's average value in this interval.
BeginX	**13**	**XBegin**	Shows cursor's initial position.
BeginY	**17**	**YBegin**	Shows the interpolated point where initial cursor meets a Pen.
DiffX	**15**	**DiffX**	Shows the difference between initial and final cursors.

OPTION	VALUE	NAME	DESCRIPTION
DiffY	**16**	**DiffY**	Shows the difference between initial and final interpolated points on the Y Axis.
EndX	**14**	**XEnd**	Shows cursor's final position.
EndY	**18**	**YEnd**	Shows the interpolated point where final cursor meets a Pen.
MaximumY	**12**	**MaxY**	Shows Pen's maximum value in this interval.
MinimumY	**11**	**MinY**	Shows Pen's minimum value in this interval.
Pen color	**6**	**Color**	Shows Pen's color.
Pen description	**5**	**Description**	Shows Pen's **DocString** property.
Pen name	**0**	**Name**	Shows Pen's name.
Status	**7**	**Status**	Shows Pen's status.
Unit	**19**	**EU**	Shows Pen's engineering unit.
X tag name	**1**	**TagX**	Shows Tag's name linked to the X Axis.
X tag value	**3**	**TagXValue**	Shows search value on X Axis.
XScale	**8**	**ScaleX**	Shows X Axis' name linked to the Pen.
Y tag name	**2**	**TagY**	Shows Tag's name linked to the Y Axis.
Y tag value	**4**	**TagYValue**	Shows search value on Y Axis.
YScale	**9**	**ScaleY**	Shows Y Axis' name linked to the Pen.

4.6.7.1.1 ChangeColumnPos

ChangeColumnPos(*Source*, *Dest*)

Swaps two columns. This method has the following parameters:

- **Source**: Index of the column to swap to *Dest*
- **Dest**: Index of the column to swap to *Source*

Example:

```
Sub CommandButton1_Click()
```

```
  Screen.Item("E3Chart1").Legend.ChangeColumnPos 1, 2
End Sub
```

4.6.7.1.2 Count

Count()

Returns the number of columns of a Legend.

4.6.7.1.3 InsertColumn

InsertColumn(*Col*, *Index*)

Inserts a new column on a Legend. This method has the following parameters:

- **Col**: Identifies a column to insert (please check the **Available options to identify a Legend column** table at the beginning of topic **Legend Methods**).
- **Index**: Determines the position to insert this column

Example:

```
Sub CommandButton1_Click()
  'Displays the Pen's name
  Screen.Item("E3Chart1").Legend.InsertColumn 0, 0
End Sub

Sub CommandButton1_Click()
  'Displays the Pen's color
  Screen.Item("E3Chart1").Legend.InsertColumn "Color", 0
End Sub
```

4.6.7.1.4 Item

Item(*Col*)

Returns a Legend's column by name or by index. The *Col* parameter determines a column's index or name (please check the **Available options for column identification** table at the beginning of topic **Legend Methods**).

4.6.7.1.5 RemoveColumn

RemoveColumn(*Col*)

Removes a column. This method has the *Col* parameter, which determines a column to remove (please check the **Available options for column identification** table at the beginning of topic **Legend Methods**).

4.6.7.2 Properties

This section contains information about the properties of the Legend object.

4.6.7.2.1 BackColor

Configures or returns a background color for a Legend. Default value of this property is white (**RGB(255, 255, 255)**).

4.6.7.2.2 EnableTextColor

This property, when enabled, specifies that a Legend's text has the same color of a Pen, configured in its **Color** property. Default value of this property is False.

4.6.7.2.3 LegendPos

Indicates a Legend's position on an E3Chart. The available options are described on the next table.

Available options for LegendPos

OPTION	DESCRIPTION
0 - lgTop	Displays a Legend on top.
1 - lgLeft	Displays a Legend on the left.
2 - lgBottom	Displays a Legend at the bottom.
3 - lgRight	Displays a Legend on the right.

4.6.7.2.4 ShowAllPens

When this property is set to True, all E3Chart Pens are displayed on a Legend. The Pen's **Visible** property is ignored. When set to False, only Pens with their **Visible** property set to True are displayed. Example:

```
Sub CommandButton1_Click()
  Set E3Chart1 = Screen.Item("E3Chart1")
  E3Chart1.Legend.ShowAllPens = Not _
    E3Chart1.Legend.ShowAllPens
End Sub
```

4.6.7.2.5 ShowHeader

Determines the visibility of a Legend's title (header). If this property is set to True, Legend's title is displayed. Otherwise, its title remains invisible.

4.6.7.2.6 Size

Determines Legend's size. This size may be its height or width, depending on Legend's position.

4.6.7.2.7 Visible

◩ Determines Legend's visibility. If this property is set to True, this Legend is visible on an E3Chart. Otherwise, it remains invisible.

4.6.7.3 Legend Columns

This section contains information about **properties** of the Legend Column object. This object does not have events nor methods associated to it.

4.6.7.3.1 Properties

This section contains information about the properties of the Legend Column object.

4.6.7.3.1.1 Caption

A Allows defining a title for a Column.

4.6.7.3.1.2 Column

Returns a Column identifier. Please check the **Available options to identify a Legend column** table, at the beginning of topic **Legend Methods**.

4.6.7.3.1.3 Format

A Configures a format used in a Column.

4.6.7.3.1.4 Index

9 Returns a Column's position on a Legend.

4.6.7.3.1.5 Name

A Determines a Column's name. Column names can be checked on the **Available options to identify a Legend column** table, at the beginning of topic **Legend Methods**.

4.6.7.3.1.6 TextAlign

Returns the text alignment of a Column. The available options for this property are described on the next table.

Available options for TextAlign

OPTION	DESCRIPTION
0 - taLeft	Left alignment

OPTION	DESCRIPTION
1 - taRight	Right alignment
2 - taCenter	Centered alignment

4.6.7.3.1.7 Width

Returns a Column's width.

4.7 E3Playback

This section contains information about **methods** and **properties** of the E3Playback object. This object does not have events associated to it.

4.7.1 Methods

This section contains information about the methods of the E3Playback object.

4.7.1.1 Pause

Pause()

Pauses the playback clock at the current time.

4.7.1.2 Play

Play()

Starts playing data at the current playback time, by moving the clock forward based on the current playing speed.

4.7.1.3 Stop

Stop()

Interrupts playback, Tag data and alarms are removed from the Screen, and no new query is performed on configured database.

4.7.2 Properties

This section contains information about the properties of the E3Playback object.

4.7.2.1 CurrentTime

Displays current date and time of this E3Playback object. This property is only available at run time.

4.7.2.2 DBServer

A Indicates the name of the Database that contains application's historical data.

4.7.2.3 InitialScreen

A Indicates the Screen that is initially displayed on E3Playback. If this property is left blank, then it uses the Screen or Frame configured in Viewer's **InitialScreen** property.

4.7.2.4 PlaybackState

Informs the status of this E3Playback object. This property is only available at run time and its options are described on the next table.

Options for PlaybackState property

OPTION	DESCRIPTION
0 - Stopped	Playback clock is stopped.
1 - Playing	Playback clock is moving.
2 - Paused	Playback clock is paused.

4.8 Reports

This section contains information about **events**, **methods**, and **properties** of the Report object.

4.8.1 Events

This section contains information about the events of the Report object.

4.8.1.1 OnAfterPrint

OnAfterPrint()

Triggered after a Section has been assembled in a Report. Users can use this event to update any necessary counter after finishing the Report.

4.8.1.2 OnBeforePrint

OnBeforePrint()

Triggered before a Section has been assembled in a Report. Users can use this event to modify an object's value in a Report before printing it. It is not recommended accessing Report's Query fields while this event is in use.

4.8.1.3 OnDataInitialize

OnDataInitialize()

Triggered before an **OnReportStart** event. This event allows adding and configuring fields to Report's **Fields** collection, before generating them. Example:

```
Sub OnDataInitialize()
  Fields.Add "Name"
  Fields.Add "Sector"
  Fields.Add "Code"
End Sub
```

4.8.1.4 OnError

OnError(*Number*, *Description*, *SCode*, *Source*, *HelpFile*, *HelpContext*, *CancelDisplay*)

Triggered by a Report's internal error. If this event is not handled, E3 then displays a generic error message.

Parameters of an OnError event

NAME	DESCRIPTION
Number	Integer identifying this error
Description	**String** with this error's description
SCode	Integer with the error code from the OLE subsystem (not used)
Source	**String** with the object that generated this error
HelpFile	**String** with the name and path of a help file
HelpContext	Number of the help topic's context referring to this error (integer)
CancelDisplay	**Boolean** indicating whether this error must be displayed on a message box

4.8.1.5 OnFetchData

OnFetchData(*eof*)

Triggered every time a new record is processed. This event is used to run a script that modifies field values added to a Report in a script linked to the **OnDataInitialize** event. Default value of the *eof* parameter is True and indicates that after the script the processing of Report's current record was finished.

4.8.1.6 OnFormat

OnFormat()

Triggered after data is read and loaded in a Report, but before a Section is prepared

for printing. This event can be used to modify the layout of a Report Section or any other object.

4.8.1.7 OnHyperlink

OnHyperlink(*Button*, *Link*)

Triggered when clicking a link in a Report. Users can use this event to run a script that redirects a link or to configure a link in a Report. The *Button* parameter indicates which button was clicked (usually 1) and the *Link* parameter determines the link's destination address.

4.8.1.8 OnNoData

OnNoData()

Triggered when there is no data to print in a Report. Users can use this event to perform a script that displays an error message on screen, warning that there is no data to print and then canceling a Report.

4.8.1.9 OnPageEnd

OnPageEnd()

Triggered at the end of the printing process of each Report's page.

4.8.1.10 OnPageStart

OnPageStart()

Triggered at the beginning of the printing process of each Report's page.

4.8.1.11 OnPrintProgress

OnPrintProgress(*PageNumber*)

Triggered while a Report's page is printing. The *PageNumber* variable indicates the number of the current page.

4.8.1.12 OnReportEnd

OnReportEnd()

Triggered at the end of Report's generation, after finishing the printing process.

4.8.1.13 OnReportStart

OnReportStart()

Triggered at the beginning of Report's generation, before starting the printing

process.

4.8.2 Methods

This section contains information about the methods of the Report object.

4.8.2.1 Export

Export([*ExportFilter*[, *ExportFileName*]])

Prints a Report according to the specified format. This method has an *ExportFilter* parameter, which determines a Report's filter, indicating an export format. It can assume the following options:

- **PDF**: Exports data to PDF (*Portable Document Format*) format
- **Excel**: Exports data to Excel's spreadsheet format
- **HTML**: Exports data to HTML (*Hyper Text Markup Language*) format
- **TEXT**: Exports data to text format
- **RTF**: Exports data to RTF (*Rich Text Format*) format
- **TIFF**: Exports data to TIFF (*Tag Image File Format*) format

When simply informing a filter name, as previously exposed, data is exported using properties common to each filter. Users can modify common properties of export filters, by using the **GetExportFilter** method, before exporting data. A file name must be informed in the *ExportFileName* parameter. Example:

```
Sub Button1_Click()
  Set report = Application.LoadReport("[Report3]")
  Select case Application._
    SelectMenu("PDF|Excel|HTML|RTF|Text|TIFF|Text(CSV)")
    Case 1
      Report.Export "PDF", "C:\mail\reports\report.pdf"
      MsgBox "Exported to PDF!"
    Case 2
      Report.Export "EXCEL", "C:\mail\reports\report.XLS"
      MsgBox "Exported to XLS!"
    Case 3
      Report.Export "HTML", "C:\mail\reports\report.html"
      MsgBox "Exported to HTML!"
    Case 4
      Report.Export "RTF", "C:\mail\reports\report.rtf"
      MsgBox "Exported to RTF!"
    Case 5
      Report.Export "TEXT", "C:\mail\reports\report.txt"
      MsgBox "Exported to TXT!"
    Case 6
```

```
      Report.Export "TIFF", "C:\mail\reports\report.tiff"
      MsgBox "Exported to TIFF!"
    Case 7
      Set reportFilter = report.GetExportFilter("TEXT")
      reportFilter.FileName = "C:\mail\reports\report2.txt"
      reportFilter.TextDelimiter = ","
      report.Export reportFilter
      MsgBox "Exported to TXT using a filter!"
  End Select
End Sub
```

4.8.2.2 GetExportFilter

GetExportFilter(*FilterName*)

Returns an object that specifies custom export parameters. This method has the *FilterName* parameter, which determines a Report's filter, indicating a format type for export. It can assume the following options:

- **PDF**: Exports data to PDF (*Portable Document Format*) format
- **Excel**: Exports data to Excel's spreadsheet format
- **HTML**: Exports data to HTML (*Hyper Text Markup Language*) format
- **TEXT**: Exports data to text format
- **RTF**: Exports data to RTF (*Rich Text Format*) format
- **TIFF**: Exports data to TIFF (*Tag Image File Format*) format

After retrieving a filter, all properties described on the next table can be modified.

Available properties on an export filter

PROPERTY	FILTER	DESCRIPTION
AutoRowHeight	Excel	In True (default), automatically configures row's height. In False,configures size to the largest element on a row.
BorderSpace	Excel	Minimum space between cells. Default value is 59 twips.
CreateCSSFile	HTML	If True, generates a CSS file in the directory indicated by **HTMLOutputPath**.

PROPERTY	FILTER	DESCRIPTION
DoubleBoundaries	Excel	In True, indicates that elements aligned to the right must replace the ones aligned to the left on the same column. Otherwise, leave it in False to free more space.
ExportRange	HTML	Indicates a range of pages for export. For example, "1, 2, 3-9, 14."
FaxExport	TIFF	Object that allows exporting data in RFC 1314 TIFF format.
FileName	All	Informs the name of a file to which data is exported.
GenPageBreaks	HTML	In True, places page breaks under the lowest element in each Report's page.
HTMLOutputPath	HTML	Default path for HTML files.
JPGQuality	PDF	Indicates the quality level of exported images (between 0 and 100).
MinColumnWidth	Excel	Minimum size of a column. Default value is 1011 twips.
MinRowHeight	Excel	Minimum size of a row.
MultiSheet	Excel	In True, each Report's page goes to a separate spreadsheet.
PageDelimiter	Text	Configures or returns a delimiter character between pages.
SuppressEmptyLines	Text	Removes or inserts empty lines, for layout purposes.
TextDelimiter	HTML	Configures or returns a delimiter character between texts.
TrimEmptySpace	Excel	In True, the vertical space among elements is removed. Default is False.
Unicode	Text	Determines whether text is saved in Unicode format (16 bits).
WebCacheOutput	HTML	In True, a Report is exported to the *WebCache* service. Otherwise (default), it is not exported.

4.8.2.3 Print

Print()

Prints a Report.

4.8.2.4 PrintPreview

PrintPreview([*Left*,][*Top*,][*Width*,][*Height*])

Generates a printing preview of a Report on screen. If the Report is correctly displayed on screen, returns True. If users click **Cancel** or any error occurs, returns False. The *Left* and *Top* parameters indicate the position of the printing preview, in pixels, starting from the upper left corner of the screen. The *Width* and *Height* parameters indicate the size of the printing preview on screen, in pixels or Himetric. All parameters are optional. Example:

```
Sub CommandButton1_Click()
  Set report = Application.LoadReport("[Report1]")
  Start = Application.GetObject("Data.Chart.datas").Value
  End = Application.GetObject("Data.Chart.datae").Value
  report.Item("Query1").SetVariableValue "Begin", Start
  report.Item("Query1").SetVariableValue "End", End
  report.PrintPreview()
End Sub
```

NOTES:

- This method is not available for Reports loaded using Server's **LoadReport** method.
- This method corresponds to **Print Report** Pick.
- If the *Left* and *Top* parameters are not defined, the printing preview is displayed at position (0, 0).
- If the *Width* and *Height* parameters are not defined, the printing preview is created with 500 x 500 pixels and the window is opened maximized. If only one of these dimensions is defined, width or height, the other dimension is configured with 500 pixels and the window is not opened maximized.
- Size values for this method can be informed as numbers or **Strings**. In case of numbers, they are considered as pixels. In case of **Strings**, if they are followed by an "hm" unit, they are interpreted as Himetric. Any other case is considered as pixels.

4.8.2.5 Query

Query()

Returns a Query object currently selected in a Report. For more information about this object, please check the **Query** chapter. Example:

```
Sub Rect_Click()
  Set Query = Application.LoadReport("[Report3]").Query()
  Query.SetVariableValue("Key1", "XYZ")
```

```
End Sub
```

4.8.3 Properties

To create a script in a Report, use the Report's Script Editor, opened by clicking **Scripts Editor** and to view a Report, click **Generate Report**. Both options are available on the **Reports** toolbar.

Report scripts use some procedures, depending on the object or Section where users want to include code. For example:

```
Report.Sections("PageHeader").Controls("E3Chart1")._
  GridBkColor= RGB(255, 0, 255)
```

Where:

- **PageHeader**: This is the name of the Section where this object is inserted in a Report
- **E3Chart1**: This is the name of the object that is inside the specified Section, in this case **PageHeader**
- **GridBkColor**: This is the object's property name, in this case **E3Chart1**
- **RGB (255, 0, 255)**: This is the property's parameter or action. In this case, changing chart's background color to pink

Therefore, to create a script in a Report, use the following concept:

```
Report.Sections("SectionName").Controls("ObjectName")._
  PropertyName = property_parameters
```

NOTE: The Report object encapsulates an ActiveReport (or AR) object, which is the Report itself.

4.8.3.1 Caption

A Contains the Report's title that appears on the title bar of the preview window. Default value of this property is an empty **String**.

4.8.4 Layout

This section contains information about **properties** of the Report's Layout object. This object does not have events nor methods associated to it.

4.8.4.1 Properties

This section contains information about the properties of the Report's Layout object.

NOTE: The properties described here are part of the ActiveReport (AR) object, which is encapsulated in the Report. These properties are only valid inside the AR scope and cannot be accessed from outside this object.

4.8.4.1.1 _PageBottomMargin

9 Determines a Report's bottom margin, in twips (one twip is equal to 1/440 inches). Default value of this property is 1440 (one inch or 2.54 cm).

4.8.4.1.2 _PageLeftMargin

9 Determines a Report's left margin, in twips (one twip is equal to 1/440 inches). Default value of this property is 1440 (one inch or 2.54 cm).

4.8.4.1.3 _PageRightMargin

9 Determines a Report's right margin, in twips (one twip is equal to 1/440 inches). Default value of this property is 1440 (one inch or 2.54 cm).

4.8.4.1.4 _PageTopMargin

9 Determines a Report's top margin, in twips (one twip is equal to 1/440 inches). Default value of this property is 1440 (one inch or 2.54 cm).

4.8.4.1.5 AllowSplitters

◩ Allows a Report's viewing area to be separated into two parts. This property is only available at run time. If this property is set to False (default value), the split bar does not appear on screen.

4.8.4.1.6 documentName

A Determines a document's name for a Report. This name appears on the print manager, and can be used to easily identify this Report. Default value of this property is "ActiveReports Document".

4.8.4.1.7 MaxPages

9 Establishes the maximum amount of pages for a Report. When this number is reached, E3 stops processing a document. Default value of this property is 10.

4.8.4.1.8 ParentReport

A This property is a variable for internal usage and contains a reference to a Report object. This is a read-only property, and valid only for the **OnDataInitialize** and **OnReportEnd** events.

4.8.4.1.9 PrintWidth

9 Determines the width of a Report's printing area, in twips. If Report's size changes at run time, its printing width must also be changed, to ensure that this Report uses the whole printing area. The size of the printing area must also include the width of margins, so that this Report do not oversize the paper size. If this happens, this error appears as a red-dotted line printed on every Report page.

4.8.4.1.10 RulerVisible

◩ When set to True, indicates that one vertical and one horizontal ruler are displayed in Report's preview window. Otherwise, these rulers remain invisible.

4.8.4.1.11 ScriptDebuggerEnabled

◩ Enables or disables the ActiveReport debugger (JIT), to debug scripts linked to Reports. This debugger is not available in E3, only in Reports.

4.8.4.1.12 ScriptLanguage

A Indicates the programming language used to interpret scripts linked to a Report. Default language is VBScript, but JScript may also be used.

4.8.4.1.13 ShowParameterUI

◩ Enables or disables Query's dialog box parameters, which appear when a Report is in execution. If this property is set to True, Query's dialog box parameters are displayed. Otherwise, these parameters are not displayed.

4.8.4.1.14 Status

Returns a Report status. The available options for this property are described on the next table.

Available options for the Status property

OPTION	DESCRIPTION
0 - DDStatIdle	Indicates that the Report is closed
1 - DDStartRunning	Indicates that the Report is in execution
2 - DDStartCompleted	Indicates that the Report is completed
3 - DDStartCanceled	Indicates that the Report was canceled

4.8.4.1.15 TOCEnabled

◩ Enables or disables a Report's table of contents. If this property is set to True, the Report's table of contents is enabled. Otherwise, the Report has no table of contents.

Default value of this property is True.

4.8.4.1.16 TOCVisible

◩ Determines the visibility of Report's table of contents. If this property is set to True, the Report's table of contents is displayed. Otherwise, the table of contents remains invisible. Default value of this property is True.

4.8.4.1.17 ToolbarVisible

◩ Enables or disables a toolbar on Report's print preview window. If this property is set to True, the toolbar is enabled. Otherwise, there is no toolbar on this window.

4.8.4.1.18 UserData

A Configures or returns specific user information. This property is similar to Visual Basic's **Tag** property, but it is exported and saved to an .rpx file. It can be used to save and load any customized information needed to draw a Report.

4.8.4.1.19 Version

A Returns a Report's version number.

4.8.4.1.20 WaterMark

Adds a background image to a Report (a watermark). Watermarks are texts or images that appear below a document's text. Usually, watermarks turn a document visually more interesting.

Example of a watermark

Default value of this property is empty (no text or image).

4.8.4.1.21 WaterMarkAlignment

Determines the watermark's alignment in the Report. The available options for this property are described on the next table.

Available options for the WaterMarkAlignment property

OPTION	DESCRIPTION
0 - ddPATopLeft	Aligns the image to the top and left
1 - ddPATopRight	Aligns the image to the top and right
2 - ddPACenter	Aligns the image to the center (default)
3 - ddPABottomLeft	Aligns the image to the bottom and left
4 - ddPABottomRight	Aligns the image to the bottom and right

4.8.4.1.22 WaterMarkPrintOnPages

Indicates the number of Report pages that receive a watermark. The syntax used may include a single page, a page range, or even a combination of both. For example: 1, 5-8, 9, 10-15.

4.8.4.1.23 WaterMarkSizeMode

Configures the watermark's size effect in a Report page. The available options are described on the next table.

Available options for the WaterMarkSizeMode property

OPTION	DESCRIPTION
0 - ddSMClicp	The watermark is displayed in the Report in its original size
1 - ddSMStretch	The watermark fills in the whole Report's page
2 - ddSMZoom	The watermark is resized up to the Report's page size

4.8.5 Section

This section contains information about common **properties** of the Report's Section object. This object does not have events nor methods associated to it.

4.8.5.1 Common Properties

This section contains information about the common properties of the Report's Section object.

4.8.5.1.1 BackColor

Specifies a background color for a Report's Section. The effect of this property is only visible if the **BackStyle** property is enabled for the option **1 - ddBKNormal**. Default value of this property is white (**RGB(255, 255, 255)**).

4.8.5.1.2 BackStyle

Specifies a background style for a Report's Section. The available options for this property are the following:

- **0 - ddBKTransparent**: Transparent background
- **1 - ddBKNormal**: Normal background

4.8.5.1.3 CanGrow

Determines the application of a *stretch* effect to the text of a Report's page. In case the page's width or height are enlarged, the text accompanies this variation. If this property is set to True, the text accompanies object's height and width variations. Otherwise, the object remains with its initial settings. Default value of this property is True.

4.8.5.1.4 CanShrink

◩ Determines the application of a *shrink* effect to the text of a Report's page. In case the page's width or height are diminished, the text accompanies this variation. If this property is set to True, the text accompanies the object's height and width variations. Otherwise, the object remains with its initial settings. Default value of this property is True.

4.8.5.1.5 height

9 Determines the page height of a Report's Section. Default value of this property is 360.

4.8.5.1.6 IsRepeating

◩ Determines a repetition of a Section on the last page of a Report. If this property is set to True, this Section is repeated on the last page. Otherwise, there is no repetition.

4.8.5.1.7 Name

A Indicates the name of a Report's Section.

4.8.5.1.8 Type

Returns the type of a Section. The available options for this property are described on the next table.

Available options for the Type property

OPTION	DESCRIPTION
0 - ReportHeader	**ReportHeader**-type Section
1 - ReportFooter	**ReportFooter**-type Section
2 - PageHeader	**PageHeader**-type Section
3 - PageFooter	**PageFooter**-type Section
4 - GroupHeader	**GroupHeader**-type Section
5 - GroupFooter	**GroupFooter**-type Section
6 - Detail	**Detail**-type Section (area for Report's content)

4.8.5.1.9 Visible

◩ Enables or disables the visibility of a Report's Section. If this property is set to True, the Section is visible on a Report. Otherwise, it is not visible. Default value of this property is True.

4.8.5.2 Group Header

This section contains information about **properties** of the Report's Group Header object. This object does not have events nor methods associated to it.

4.8.5.2.1 Properties

This section contains information about the properties of the Report's Group Header object.

4.8.5.2.1.1 ColumnLayout

Determines whether a **GroupHeader** is using the same layout of columns configured in the **Detail** Section. If this property is set to True, the number of columns in the **Detail** Section is the same as the associated **GroupHeader** or **GroupFooter**. Otherwise, it remains with its standard configuration.

4.8.5.2.1.2 DataField

Returns data from Report fields. Defines a mandatory field for a group inside the content of the **Detail** Section. This value is set to the name of all fields in the Report's data source, or to the name of a custom field inserted in the fields collection. When this property is set, the Report creates a new group every time a field value changes in the data records of a detail.

4.8.5.2.1.3 GrpKeepTogether

Determines if a **GroupHeader** Section prints as a single block on the same Report's page. The available options for this property are described on the next table.

Available options for the GrpKeepTogether property

OPTION	DESCRIPTION
0 - GrpNone	The page can be broken immediately after a **GroupHeader**
1 - GrpFirstDetails	The **GroupHeader** prints with the first **Detail** Section of the same Report's page or column
2 - GrpAll	The **GroupHeader**, **Detail**, and **Footer** of a group are printed together in the same Report's page

Default value of this property is **0 - GrpNone**.

4.8.5.2.1.4 KeepTogether

Determines whether Report Sections are printed as a single block on the same page. The available options for this property are described on the next table.

Available options for the KeepTogether property

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report
1 - ddGrpFirstDetail	The Report prints a **Detail** Section on the same page or column

4.8.5.2.1.5 NewColumn

Inserts a new column break before or after printing a Section in the Report. The available options for this property are described on the next table.

Available options for the NewColumn property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section
1 - ddNBefore	Starts printing a Section on a new page
2 - ddNPAfter	Starts a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing on a new page and a new page after printing a Section

4.8.5.2.1.6 NewPage

Inserts a page break in the Report. The available options for this property are described on the next table.

Available options for the NewPage property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in the Section (default)
1 - ddNBefore	Starts printing on a new page
2 - ddNPAfter	Starts printing a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing on a new page and starts a new page after printing a Section.

4.8.5.2.1.7 Repeat

Determines whether a **GroupHeader** is printed again after linked to a **Detail** Section when there are multiple pages, columns, or page breaks in the Report. The available options for this property are described on the next table.

Available options for the Repeat property

OPTION	DESCRIPTION
0 - ddRepeatNone	There is no re-printing of a group header (default)
1 - ddRepeatOnPage	Prints group headers on top of the page according to the specifications of the **Detail** Section
2 - ddRepeatOnColumn	Prints group headers on top of the Report's page column according to the specifications of the **Detail** Section
3 - ddRepeatAll	Prints group headers and other objects on top of a Report's page according to the specifications of the **Detail** Section

4.8.5.2.1.8 UnderlayNext

Determines whether a Section must print one Section after another, consecutively. If this property is set to True, the next Section starts printing from the Section's upper coordinate on a Report's page. Otherwise, this resource is not used. Default value of this property is False.

4.8.5.3 Detail

This section contains information about **properties** of the Report's Detail object. This object does not have events nor methods associated to it.

4.8.5.3.1 Properties

This section contains information about the properties of the Report's Detail object.

4.8.5.3.1.1 ColumnCount

Determines the number of columns of a Report's **Detail** Section. The width of each column must be equal to Report's printable area, divided by the number of columns. Default value of this property is 1 (one).

4.8.5.3.1.2 ColumnDirection

The **ColumnDirection** property determines the printing direction of columns in a **Detail** Section. The available options for this property are described on the next table.

Available options for the ColumnDirection property

OPTION	DESCRIPTION
0 - ddCDDownAcross	Prints each column of a **Detail** Section from top to bottom and then moves to the next column on the right
1 - ddCDAcrossDown	Prints each column of a **Detail** Section from right to left, and so on

The layout is determined according to the configured option, as shown on the next figure.

ddCDDownAccross option

ddCDAccrossDown option

4.8.5.3.1.3 ColumnSpacing

Determines the spacing of columns on a **Detail** Section. Default value of this property is 0 (zero).

4.8.5.3.1.4 KeepTogether

Determines whether Report Sections are printed as a single block on the same page. The available options for this property are described on the next table.

Available options for the KeepTogether property

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report
1 - ddGrpFirstDetail	The Report prints a **Detail** Section on the same page or column

4.8.5.3.1.5 NewColumn

Inserts a new column break before or after printing a Section in the Report. The available options for this property are described on the next table.

Available options for the NewColumn property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in a Section

OPTION	DESCRIPTION
1 - ddNBefore	Starts printing a Section on a new page
2 - ddNPAfter	Starts a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing on a new page and a new page after printing a Section

4.8.5.3.1.6 NewPage

Inserts a page break in the Report. The available options for this property are described on the next table.

Available options for the NewPage property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in a Section
1 - ddNBefore	Starts printing on a new page
2 - ddNPAfter	Starts a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing on a new page and starts a new page after printing a Section

4.8.5.4 Group Footer

This section contains information about **properties** of the Report's Group Footer object. This object does not have events nor methods associated to it.

4.8.5.4.1 Properties

This section contains information about the properties of the Report's Group Footer object.

4.8.5.4.1.1 ColumnLayout

Determines whether a **GroupFooter** uses the same column layout configured on a **Detail** Section. If this property is set to True, the number of columns of a **Detail** Section reflects in the linked **GroupHeader** or **GroupFooter**. Otherwise, the default configuration remains. Default value of this property is True.

4.8.5.4.1.2 KeepTogether

Determines whether Report Sections are printed as a single block on the same page. The available options for this property are described on the next table.

Available options for the KeepTogether property

OPTION	DESCRIPTION
0 - ddGrpNone	There is a page break after the Report
1 - ddGrpFirstDetail	The Report prints a **Detail** Section on the same page or column

4.8.5.4.1.3 NewColumn

Inserts a new column break before or after printing a Section in the Report. The available options for this property are described on the next table.

Available options for the NewColumn property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break on a Section
1 - ddNBefore	Starts printing a Section on a new page
2 - ddNPAfter	Starts a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing on a new page and a new page after printing a Section

4.8.5.4.1.4 NewPage

Inserts a page break in the Report. The available options for this property are described on the next table.

Available options for the NewPage property

OPTION	DESCRIPTION
0 - ddNPNone	There is no page break in a Section (default)
1 - ddNBefore	Starts printing on a new page
2 - ddNPAfter	Starts printing a new page after printing a Section
3 - ddNPBeforeAfter	Starts printing a new page and starts a new page after printing a Section

4.8.5.4.1.5 PrintAtBottom

Determines whether the **Group Footer** or a Report's footer are printed at a page's bottom. If this property is set to True and the Report has a page footer, the **Group Footer** and the Report footer are printed above the **Footer** Section of a page. Configuring more than one Section to print the Report's footer allows that the following footer's Sections print on separate pages.

4.8.6 Objects

This section contains information about properties of Report objects. These objects do not have events nor methods associated to them.

4.8.6.1 Common Properties

This section contains information about common properties of the Report objects.

4.8.6.1.1 BackColor

Specifies a background color for an object in the Report. The effect of this property is only visible if the **BackStyle** property is configured with the **1 - ddBKNormal** option. Default value of this property is white (**RGB(255, 255, 255)**).

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.1.2 BackStyle

Specifies a background style for objects in the Report. The available options for this property are the following:

- **0 - ddBKTransparent**: Transparent (displays the color defined by Section's **BackColor** property)
- **1 - ddBKNormal**: Normal (displays the color defined by object's **BackColor** property)

Default value of this property is **0 - ddBKTransparent**.

NOTE: This property is not valid for Line, Barcode, Page Break, and Frame objects.

4.8.6.1.3 height

9 This property determines the height of an object in the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.1.4 left

9 Returns the object's left positioning value in the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.1.5 Name

A Determines the name of an object. Default value of this property is an empty **String**.

4.8.6.1.6 Tag

A Returns the type of a Tag linked to an object, that is, **Boolean**, **String**, **Integer**, etc.

4.8.6.1.7 Top

Returns the value of an object's top.

NOTE: This property is not valid for a Line object.

4.8.6.1.8 Visible

Determines the visibility of an object in the Report. If this property is set to True, this object is visible in the Report. Otherwise, this object is hidden. Default value of this property is True.

NOTE: This property is not valid for a Page Break object.

4.8.6.1.9 Width

This property determines the width of an object in the Report.

NOTE: This property is not valid for Line and Page Break objects.

4.8.6.2 Barcode

This section contains information about **properties** of the Report's Barcode object. This object does not have events nor methods associated to it.

4.8.6.2.1 Properties

This section contains information about the properties of the Report's Barcode object.

4.8.6.2.1.1 Alignment

Determines the text's alignment of a Barcode object in the Report. The available options are the following:

- **0 - ddtxLeft**: Aligns text to the left of the object
- **1 - ddtxRight**: Aligns text to the right of the object
- **2 - ddtxCenter**: Aligns text to the center of the object

4.8.6.2.1.2 BarWidth

Determines the width of Barcode's bars. Configuring this width to 1 (one), the object's bar is expanded in up to 15 points, and so on. The higher the number

configured in this property, the larger the width of Barcode's bar. Default value of this property is 1 (one).

4.8.6.2.1.3 Caption

A Contains the text of the Barcode object itself. Default value of this property is an empty **String**.

4.8.6.2.1.4 CaptionPosition

Specifies the position of the text in the **Caption** property of a Barcode object. The available options for this property are the following:

- **0 - ddbcCaptionNone**: Text of the **Caption** property does not appear in the Barcode
- **1 - ddbcCaptionAbove**: Text of the **Caption** property appears above the Barcode
- **2 - ddbcCaptionBelow**: Text of the **Caption** property appears below the Barcode

Default value of this property is **0 - ddbcCaptionNone**.

4.8.6.2.1.5 DataField

A Configures or returns data linked to a Barcode object. This linked data may be a field from a Database table provided by a Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by an equal symbol), or an E3 Tag or property. In this case, the current variable's value is displayed when printing. Default value of this property is an empty **String**.

NOTE: The server must be executing so that a variable's value can be captured.

4.8.6.2.1.6 Direction

Determines the spatial orientation of Barcodes. The available options for this property are described on the next table.

Available options for the Direction property

OPTION	DESCRIPTION
0 - ddbcLeftToRight	Barcode's orientation is from left to right
1 - ddbcRightToLeft	Barcode's orientation is from right to left
2 - ddbcTopToBottom	Barcode's orientation is from top to bottom
3 - ddbcBottomToTop	Barcode's orientation is from bottom to top

Default value of this property is **0 - ddbcLeftToRight**.

4.8.6.2.1.7 EnableCheckSum

Enables or disables the reading of a checksum value (a character from a Barcode object). If this property is set to False, only codes with a checksum are affected.

4.8.6.2.1.8 Font

Determines the font of a Barcode object in the text established by the **Caption** property. Default value of this property is "Arial".

NOTE: This property cannot be used on scripts or Links, being edited only via Studio.

4.8.6.2.1.9 Forecolor

Specifies a foreground color for a Barcode object. In scripts, use VBScript's **RGB** method to build a color to link to this property. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.2.1.10 Style

Determines a style for a Barcode. The available options for this property are described on the next table.

Available options for the Style property

OPTION	DESCRIPTION
0 - ddbcNone	Default Barcode style
1 - ddbcAnsi39	ANSI 3 of 9 (Code 39). Use letters, numbers, -,*, $, /, +, %, etc.
2 - ddbcAnsi39x	Extended ANSI 3 of 9 (Extended Code 39). Use full ASCII characters
3 - ddbcCode_2_of_5	2 of 5. Use only numbers
4 - ddbcCode25intlv	Interleaved 2 of 5. Use only numbers
5 - ddbcCode25mat	Matrix 25
6 - ddbcCode39	Code 39. Use letters, numbers, -, *, $, /, +, %, etc.
7 - ddbcCode39x	Extended Code 39. Use full ASCII characters.
8 - ddbcCode_128_a	128 A. Use numbers, punctuation, or letters
9 - ddbcCode_128_b	128 B. Use strings, numbers, punctuation, or letters
10 - ddbcCode_128_c	128 C. Use only numbers.

OPTION	DESCRIPTION
11 - ddbcCode_128auto	128 Automatic. Use full ASCII characters. Automatically selects codes between 128 A, B, and C to set the smallest Barcode value
12 - ddbcCode_93	Code 93. Use letters, numbers, -, *, $, /, +, %, etc.
13 - ddbcCode_93x	Extended Code 93. Use full ASCII characters
14 - ddbcMSI	MSI Code. Use only numbers
15 - ddbcPostNet	PostNet. Use only numbers with a digital verification
16 - ddbcCodabar	Codabar. Use A, B, C, D, +, -, :, /, or numbers
17 - ddbcEAN_8	EAN-8. Use only numbers (seven numbers and digital verification)
18 - ddbcEAN_13	EAN-13. Use only numbers (12 numbers and digital verification)
19 - ddbcUPC_A	UPC-A. Use only numbers (11 numbers and digital verification)
20 - ddbcUPC_EO	UPC-E1. Use only numbers. Used for UPC zero-compression symbols. In the **Caption** property, users can enter six digits of a UPC-E code or 11 digits. If an 11 digits code is inserted, the Barcode converts it to UPC-E six digits, if possible. Otherwise, converts from 11 to six UPC-E digits and nothing is displayed.
21 - ddbcUPC_E1	UPC-E1. Use only numbers. The width of input data of UPC E1 is six number characters
22 - ddbcRM4SCC	Royal Mail RM4SCC. Use only letters and numbers (with digital verification). This Barcode is used in Great Britain
23 - ddbcUCCEAN128	UCC/EAN_128. Use full ASCII characters. The special version of the Code 128 is used on HIBC application

4.8.6.3 Ellipse, Rectangle, and Round Rectangle

This section contains information about **properties** of the Report's Ellipse, Rectange, and Round Rectangle objects. These objects do not have events nor methods associated to them.

4.8.6.3.1 Properties

This section contains information about the properties of the Report's Ellipse, Rectangle and Round Rectangle objects.

4.8.6.3.1.1 LineColor

Specifies a color for Elipse, Rectangle, and Round Rectangle objects' line. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.3.1.2 LineStyle

Determines a style for Ellipse, Rectangle, and Round Rectangle objects' line. The available options for this property are described on the next table.

Available options for the LineStyle property

OPTION	DESCRIPTION
0 - ddLSTransparent	Object's line becomes transparent
1 - ddLSSolid	Object's line appears solid
2 - ddLSDash	Object's line appears dashed
3 - ddLSDot	Object's line appears dotted
4 - ddLSDashDot	Object's line appears with dashes and dots
5 - ddLSDashDotDot	Object's line appears dashed, dotted, and dotted

Default value of this property is **1 - ddLSSolid**.

4.8.6.3.1.3 LineWeight

Specifies a width for Ellipse, Rectangle, and Round Rectangle objects' line. By configuring this width to 1 (one) the object's line is expanded in up to 15 points, if configured to 2 (two) the object's line is expanded in up to 30 points, and so on. The higher the number configured in this property, the larger the object's width. Default value of this property is 1 (one).

4.8.6.3.1.4 Shape

Allows changing the shape of Ellipse, Rectangle, and Round Rectangle objects. The available options for this property are described on the next table.

Available options for the Shape property

OPTION	DESCRIPTION
0 - ddSHRectangle	Rectangular shape
1 - ddSHEllipse	Elliptical or circular shape
2 - ddSHRoundRect	Round rectangle shape

4.8.6.4 Picture

This section contains information about **properties** of the Report's Picture object. This object does not have events nor methods associated to it.

4.8.6.4.1 Properties

This section contains information about the properties of the Report's Picture object.

4.8.6.4.1.1 DataField

A Configures or returns data associated to a Picture object. This associated data may be a field in a Database table provided by a Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by the equal symbol), or an E3 Tag or property. In this case, the current value of this variable is displayed when printing. Default value of this property is an empty **String**.

NOTE: The server must be executing so that this variable's value can be captured.

4.8.6.4.1.2 Forecolor

Specifies a foreground color for a Picture object. In scripts, use VBScript's **RGB** method to build a color to link to this property. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.4.1.3 hyperLink

A Specifies a *link* set to a text. To use this resource, use the **OnHyperLink** event. Default value of this property is an empty **String**.

4.8.6.4.1.4 LineColor

Specifies a color for a Picture object's line. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.4.1.5 LineStyle

Determines a style for a Picture object's line. The available options for this property are described on the next table.

Available options for the LineStyle property

OPTION	DESCRIPTION
0 - ddLSTransparent	Object's line becomes transparent

OPTION	DESCRIPTION
1 - ddLSSolid	Object's line appears solid.
2 - ddLSDash	Object's line appears dashed
3 - ddLSDot	Object's line appears dotted
4 - ddLSDashDot	Object's line appears with dashes and dots
5 - ddLSDashDotDot	Object's line appears dashed, dotted, and dotted

Default value of this property is **1 - ddLSSolid**.

4.8.6.4.1.6 LineWeight

9 Specifies the width of a Picture object's line. By configuring this width to 1 (one) the Picture object's line expands in up to 15 points, and so on. The higher the number configured in this property, the larger the Picture object's width. Default value of this property is 1 (one).

4.8.6.4.1.7 Picture

A Specifies an image file for a Picture object. Allowed extensions are .bmp, .gif, .jpg, .cur, .ico, .emf, and .wmf. Default value of this property is an empty **String**.

4.8.6.4.1.8 PictureAlignment

Determines the alignment of an image in the Picture object. The available options for this property are described on the next table.

Available options for the PictureAlignment property

OPTION	DESCRIPTION
0 - ddPATopLeft	Aligns the image on object's top left
1 - ddPATopRight	Aligns the image on object's top right
2 - ddPACenter	Aligns the image on object's center
3 - ddPABottomLeft	Aligns the image on object's bottom left
4 - ddPABottomRight	Aligns the image on object's bottom right

Default value of this property is **2 - ddPACenter**.

4.8.6.4.1.9 SizeMode

Specifies the size of a Picture object. The available options for this property are described on the next table.

Available options for the SizeMode property

OPTIONS	DESCRIPTION
0 - ddsMClip	Displays a Picture object in its current size
1 - ddsMStretch	Adjusts a Picture object according to its area
2 - ddsMZoom	Adjusts the height or width of an image in the Picture object inside the specified area, without distorting it

4.8.6.5 SetPoint

This section contains information about **properties** of the Report's SetPoint object. This object does not have events nor methods associated to it.

4.8.6.5.1 Properties

This section contains information about the properties of the Report's SetPoint object.

4.8.6.5.1.1 Alignment

Determines the alignment of a text in a SetPoint object. The available options for this property are:

- **0 - Left**: Left alignment (default)
- **1 - Right**: Right alignment
- **2 - Center**: Center alignment

4.8.6.5.1.2 CanGrow

Determines the application of a *stretch* effect to a text in a SetPoint object. If the width or height of this object are increased, the text accompanies this variation. If this property is set to True, the text accompanies object's height and width variations. Otherwise, this object remains with its initial settings. Default value of this property is True.

4.8.6.5.1.3 CanShrink

Determines the application of a shrinking effect to a text in a SetPoint object. If the width or height of this object are decreased, the text accompanies this variation. If this property is set to True, text accompanies object's height and width variations. Otherwise, this object remains with its initial settings. Default value of this property is True.

4.8.6.5.1.4 ClassName

A Returns the class of a SetPoint object. This is a read-only property.

4.8.6.5.1.5 DataField

A Configures or returns associated data to a SetPoint object. This associated data can be a field on a Database table provided by a Query object, a mathematical expression with Query fields and VBScript functions (in this case, this field must be preceded by an equal symbol), or an E3 Tag or property. In this case the current value of this variable is displayed when printing. Default value of this property is empty.

NOTE: The server must be executing so that the variable's value can be captured.

4.8.6.5.1.6 Font

A This property determines a font for a text of a SetPoint object. Default value of this property is an empty **String**. This property cannot be used in scripts or Links and it is configured only via Studio.

4.8.6.5.1.7 ForeColor

The **ForeColor** specifies a background color for a SetPoint object. In scripts, use VBScript's **RGB** method to build a color to link to this property. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.5.1.8 hyperLink

A The **hyperLink** property determines the *link* that is set to the text. To use this resource, use the **OnHyperLink** event.

4.8.6.5.1.9 Multiline

The **Multiline** property indicates whether a text has multiple lines (True) or is a simple text box (False). This can be viewed when Viewer is executing. Default value of this property is False.

4.8.6.5.1.10 OutputFormat

A Configures or returns the format of text in the **Text** property, used on one of Visual Basic's format functions (**FormatCurrency**, **FormatDateTime**, **FormatNumber**, and **FormatPercent**).

4.8.6.5.1.11 Style

A Returns the style of a text configured in a SetPoint object. This is a read-only property.

4.8.6.5.1.12 SummaryDistinctField

A Determines the name of a field used by the selected function in the **SummaryFunc** property. This property is only valid if the function defined in the **SummaryFunc** property is from the **Distinct Summary** function group, which encompasses functions from 9 to 15, and when the **SummaryType** property is different from 0 (zero).

4.8.6.5.1.13 SummaryFunc

Determines the type of function used for processing field values specified in the **DataField** property, as described on the next table. This property is only valid when the **SummaryType** property is different from 0 (zero).

Available options for the SummaryFunc property

OPTION	DESCRIPTION
0 - Sum	Calculates the sum of all values inside the specified total intervals (group, page, or Report)
1 - Avg	Calculates the average of all values inside the specified total intervals (group, page, or Report)
2 - Count	Counts the number of all values inside the specified total intervals (group, page, or Report)
3 - Min	Displays the lowest value (minimum value) of all values inside the specified total intervals (group, page, or Report)
4 - Max	Displays the highest values (maximum value) of all values inside the specified total intervals (group, page, or Report)
5 - Var	Calculates the variance of the values inside the specified total intervals (group, page or Report)
6 - VarP	Calculates the population variance of the values inside the specified total intervals (group, page or Report)
7 - Dev	Calculates the deviation of the values inside the specified total intervals (group, page or Report)
8 - DevP	Calculates the population deviance of the values inside the specified total intervals (group, page or Report)

OPTION	DESCRIPTION
9 - DSum	Calculates the sum of all distinct values inside the specified total intervals (group, page or Report)
10 - DAvg	Calculates the average based on distinct values inside the specified total intervals (group, page or Report)
11 - DCount	Counts the number of distinct values inside the specified total intervals (group, page or Report)
12 - DVar	Calculates the variance of the distinct values inside the specified total intervals (group, page or Report)
13 - DVarP	Calculates the population variance of the distinct values inside the specified total intervals (group, page or Report)
14 - DDev	Calculates the deviance of the distinct values inside the specified total intervals (group, page or Report)
15 - DDevP	Calculates the population deviance of the distinct values inside the specified total intervals (group, page or Report)

4.8.6.5.1.14 SummaryGroup

A This property is only valid when the **SummaryType** property is equal to **3 - SubTotal**. **SummaryGroup** indicates the name of a **GroupHeader** Section used to control subtotals, that is, at every change in **GroupHeader**'s value, the sum is then restarted.

NOTE: When this property is used, the **CanShrink** and **CanGrow** properties remain disabled.

4.8.6.5.1.15 SummaryRunning

Determines if a summary of running totals is performed, according to the following options:

- **0 - None**: Does not perform this summary
- **1 - Group**: Calculates the summary of running totals for each total interval specified
- **2 - All**: Calculates the summary of running totals of all values in the Report, despite of groupings

This property is only valid when the **SummaryType** property is different from 0 (zero).

4.8.6.5.1.16 SummaryType

Determines the type or level of summary to generate. The available options for this property are described on the next table.

Available options for the SummaryType property

OPTION	DESCRIPTION
0 - None	Summaries are not generated
1 - GrandTotal	Specifies that the whole content of the Report is summarized
2 - PageTotal	Specifies that a subtotal by page is generated
3 - SubTotal	Specifies that a subtotal for each group is generated, defined in the **SummaryGroup** property
4 - PageCount	Specifies a page counter

4.8.6.5.1.17 Text

Determines a text to set to a SetPoint object.

4.8.6.5.1.18 VerticalAlignment

Determines the vertical alignment of a SetPoint object, as follows:

- **0 - Top**: Top alignment
- **1 - Middle**: Middle alignment
- **2 - Bottom**: Bottom alignment

4.8.6.5.1.19 WordWrap

Enables or disables a line break in a text, if the available area for this text overrides the limits determined by the SetPoint object. For this property to work, the **Multiline** property must be equal to True. If it is False, the **white-space:nowrap** configuration appears in the **Style** property.

4.8.6.6 Text

This section contains information about **properties** of the Report's Text object. This object does not have events nor methods associated to it.

4.8.6.6.1 Properties

This section contains information about the Report's Text object.

4.8.6.6.1.1 Alignment

The **Alignment** property determines a text's alignment, as follows:

- **0 - ddtxLeft**: Left alignment
- **1 - ddtxRight**: Right alignment
- **2 - ddtxCenter**: Center alignment (default value)

4.8.6.6.1.2 Angle

The **Angle** property indicates a text's angle. This property's value must be specified in decimal degrees, that is, for a text to be displayed in a 45-degrees angle, this value must be equal to 450. Default value of this property is 0 (zero, horizontal positioning).

4.8.6.6.1.3 Caption

The **Caption** property contains the text of the object itself. Default value of this property is an empty **String**.

4.8.6.6.1.4 ClassName

The **ClassName** property allows specifying a global CSS class (indicated by an external CSS stylesheet) to apply to a text. A CSS class (*Cascading Style Sheet*) is a format pattern that determines letter's type and size, paragraph's alignment and spacing, among other features. Using CSS, users can apply a pre-defined format pattern to a text, unifying and speeding up text presentation. To apply a specific style, users can use the **Style** property. Default value of this property is the **Normal** style.

4.8.6.6.1.5 Font

This property indicates the name of a text's font (letter type). Default value of this property is an empty **String** (E3 uses operating system's default).

NOTE: This property cannot be used in scripts or Links, it is only configured via Studio, and can only be changed at run time.

4.8.6.6.1.6 ForeColor

The **Forecolor** property specifies a foreground color for a Text object. In scripts, use VBScript's **RGB** method to build a color to link to this property. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.6.1.7 hyperLink

The **hyperLink** property determines a *link* to set to a Text object. To use this resource, use the **OnHyperLink** event. Default value of this property is an empty **String**.

4.8.6.6.1.8 Multiline

The **Multiline** property indicates whether a text contains multiple lines (True) or is a simple text box (False). This can be viewed when Viewer is executing. Default value of this property is False.

4.8.6.6.1.9 Style

The **Style** property allows specifying a CSS (*Cascade Style Sheet*) style for a text, replacing the global style. This property's value must be a valid CSS **String**, otherwise this property is ignored. Default value of this property is an empty **String** (E3 uses operating system's default). Example:

```
Sub Report1_OnBeforePrint
  Label1.Style = "font-family: Times; font-weight: bold;_
    text-align: center; color: RGB(255, 255, 0)"
End Sub
```

4.8.6.6.1.10 VerticalAlignment

The **VerticalAlignment** property determines the vertical alignment of a text, as follows:

- **0 - ddTxTop**: Top alignment (default)
- **1 - ddTxMiddle**: Center alignment
- **2 - ddTxBottom**: Bottom alignment

4.8.6.6.1.11 WordWrap

Enables or disables a line break on a text, in case the available area for a text overrides the limits determined in an object. For this property to work, the **Multiline** property must be equal to True.

4.8.6.7 Line

This section contains information about **properties** of the Report's Line object. This object does not hae events nor methods associated to it.

4.8.6.7.1 Properties

This section contains information about the properties of the Report's Line object.

4.8.6.7.1.1 LineColor

The **LineColor** property specifies a color for a Line object's line. Default value of this property is black (**RGB(0, 0, 0)**).

4.8.6.7.1.2 LineStyle

This property determines the style of an object's line. Default value of this property is **1 - ddLSSolid**. The other available options for this property are described on the next table.

Available options for the LineStyle property

OPTION	DESCRIPTION
0 - ddLSTransparent	Object's line becomes transparent
1 - ddLSSolid	Object's line appears solid
2 - ddLSDash	Object's line appears dashed
3 - ddLSDot	Object's line appears dotted
4 - ddLSDashDot	Object's line appears with dashes and dots
5 - ddLSDashDotDot	Object's line appears dashed, dotted, and dotted

4.8.6.7.1.3 LineWeight

The **LineWeight** property specifies the width of an object's line. Configuring this width to 1 (one), an object's line expands in up to 15 points, if this width is 2 (two), this object's line expands in up to 30 points, and so on. The higher the number configured in this property, the larger the object's width. Default value of this property is 1 (one).

4.8.6.7.1.4 X1

The **X1** property enables or disables the position of an initial point of X axis line.

4.8.6.7.1.5 X2

The **X2** property determines the position of the ending point of X axis line. Default value of this property is empty.

4.8.6.7.1.6 Y1

The **Y1** property determines the position of the starting point of Y axis line. Default value of this property is empty.

4.8.6.7.1.7 Y2

The **Y2** property determines the position of the ending point of Y axis line. Default value of this property is empty.

4.8.6.8 Page Break

This section contains information about the **properties** of the Report's Page Break object. This object does not have events nor methods associated to it.

4.8.6.8.1 Properties

This section contains information about the properties of the Report's Page Break object. This object does not have events nor methods associated to it.

4.8.6.8.1.1 Enabled

The **Enabled** property enables or disables a Page Break object in the Report. If this property is set to True, this object is enabled in the Report. Otherwise, this object remains disabled. Default value of this property is True.

4.8.6.9 Frame

This section contais information about **properties** of the Report's Frame object. This object does not have events nor methods associated to it.

4.8.6.9.1 Properties

This section contains information about the properties of the Report's Frame object.

4.8.6.9.1.1 CanGrow

Determines the application of a *stretch* effect to the text of a Frame object. If the width or height of this object is increased, the text accompanies this variation. If this property is set to True, the text accompanies variations in object's height and width. Otherwise, this object remains with its initial settings. Default value of this property is True.

4.8.6.9.1.2 CanShrink

◩ Determines the application of a shrinking effect to the text of a Frame object. If the width or height of this object is decreased, the text accompanies this variation. If this property is set to True, the text accompanies variations in object's height and width variations. Otherwise, this object remains with its initial settings. Default value of this property is True.

4.8.6.9.1.3 CloseBorder

◩ This property enables or disables viewing a border line of a Frame's base, if it overrides more than one page in the Report.

4.8.6.9.1.4 left

9 The **left** property returns the left position of a Frame object in the Report. Default value of this property is empty.

4.8.6.10 E3Chart

This section contains information about the **properties** of the Report's E3Chart object.

4.8.6.10.1 Properties

Usage example of E3Chart properties in a Report

The following scripts must be created in Report's **Page Header** Section, using the **OnBeforePrint** event. Examples:

```
Sub OnBeforePrint
  'Using an E3Chart in a Report
  Set chart = _
    Report.Sections("PageHeader").Controls("E3Chart1")
  chart.LoadData()
  chart.FitAll
End Sub

Sub OnBeforePrint
  ' This script copies configurations from
  ' E3Chart chartfrom object to E3Chart chart
  ' which is used in the report.
  Set chartfrom = _
    Application.GetFrame().Screen.Item("E3Chart1")
  Set chart = _
    Report.Sections("PageHeader").Controls("E3Chart2")
  chart.CopyConfig(chartfrom)
  chart.LoadData()
  chart.FitAll
```

```
End Sub
```

NOTE: Descriptions for E3Chart properties in a Report are the same explained in the topic **View - E3Chart - Properties**.

CHAPTER 5

Server Objects

This section contains information about common properties of Server objects. These objects do not have common events nor methods. Objects sharing Server properties are the following:

- **Server's Runtime Objects**
- **Configuration-Only Objects**
- **Drivers**
- **Data Server**
- **Database**
- **Historic**
- **Storage**
- **Formulas**
- **Alarms**

5.1 Common Properties

This section contains information about common properties of all Server objects.

5.1.1 IsAlarmArea

◩ Enables or disables the Alarm Area functionality for Server objects. This property is not available for the following objects: Alarm Area, Alarm Source, and Alarm Server. The default value of this property is False. When enabling this property, the object then has the same properties of an **Alarm Area**.

5.1.2 Common Properties of Server Objects as Alarm Areas

This section contains information about common properties of Server objects that behave as Alarm Areas.

5.1.2.1 ActiveAlarms

9 Determines the number of active alarms inside an object. This is a read-only property.

5.1.2.2 ActiveHighAlarms

Indicates the number of active alarms with high severity. This is a read-only property.

5.1.2.3 ActiveHighNACKAlarms

Indicates the number of non-acknowledged alarms with high severity. This is a read-only property.

5.1.2.4 ActiveLowAlarms

Indicates the number of active alarms with low severity. This is a read-only property.

5.1.2.5 ActiveLowNACKAlarms

Indicates the number of non-acknowledged alarms with low severity. This is a read-only property.

5.1.2.6 ActiveMedAlarms

Indicates the number of active alarms with medium severity. This is a read-only property.

5.1.2.7 ActiveMedNACKAlarms

Indicates the number of non-acknowledged alarms with medium severity. This is a read-only property.

5.1.2.8 ActiveNACKAlarms

Indicates the number of non-acknowledged alarms inside an object. This is a read-only property.

5.1.2.9 Alarm

Indicates that there are active alarms inside the object. If this option is set to True, there is at least one active alarm inside the object, and the property **ActiveAlarms** performs a reading from server, then indicating the amount of active alarms. Otherwise, property **ActiveNACKAlarms** performs a reading of non-acknowledged alarms. This is a read-only property.

5.1.2.10 AlarmVerify

◩ Enables a check on all alarms inside the object. After enabling that check (True), if the value of property **ActiveAlarms** is greater than 0 (zero), server then checks active alarms, as well as non-acknowledged ones, listing these last ones in the property **ActiveNACKAlarms**. This property is useful to avoid a cascade effect on some systems, where the occurrence of an event triggers a large amount of related alarms.

5.1.2.11 InactiveHighNACKAlarms

9 Indicates the number of active and unacknowledged alarms with high severity. This is a read-only property.

5.1.2.12 InactiveLowNACKAlarms

9 Indicates the number of active and unacknowledged alarms with low severity. This is a read-only property.

5.1.2.13 InactiveMedNACKAlarms

9 Indicates the number of active and unacknowledged alarms with medium severity. This is a read-only property.

5.1.2.14 InactiveNACKAlarms

9 Determines the total amount of active and unacknowledged alarms. This is a read-only property.

5.1.2.15 UserFields

🔗 Returns an object that is a collection of Alarm's User Fields of a Server object. Please check the item **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.2 Collection of Alarm's User Fields

This section contains information about **methods** and **properties** common to the collection of Alarm's User Fields of E3's Server objects, returned by the **UserFields** property of **Alarm Areas** and **Alarm Sources**.

5.2.1 Common Methods

This section contains information about methods common to the collection of Alarm's User Fields of E3's Server objects.

5.2.1.1 Item

Item(*Index*)

Returns an Alarm's User Field object indicated by the *Index* parameter, which can be the value of the **Index** property (integer) or the value of the **Name** property (text) of this object.

5.2.2 Common Properties

This section contains information about the properties common to the collection of Alarm's User Fields of E3's Server objects.

5.2.2.1 Count

9 Returns the number of child objects (items) of a collection of Alarm's User Fields. If this Collection has no child objects, then this property returns 0 (zero).

5.2.3 Alarm's User Fields

This section contains information about **properties** of objects of type Alarm's User Field in the Collection of Alarm's User Fields returned by the UserFields property of **Alarm Areas**, **Alarm Sources**, and **Server Objects** that behave as Alarm Areas. This object does not have events nor methods associated to it.

5.2.3.1 Properties

This section contains information about the properties of objects of type Alarm's User Field in the Collection of Alarm's User Fields.

5.2.3.1.1 Index

9 Returns the index of this object in the Collection of Alarm's User Fields. This value can be used as a parameter for that Collection's **Item** method.

5.2.3.1.2 Link

A Returns the Link configured for this Alarm's User Field. To change the Link configured in this property at run time, users must deactivate this object. For example:

```
Dim sAlarm
Set sAlarm = _
  Application.GetObject("ConfigAlarms.Area.DigitalAlarm1")
sAlarm.Deactivate()
sAlarm.UserFields.Item("BatchName").Link = _
  "Driver.TagBatchName.Value"
sAlarm.Activate()
```

5.2.3.1.3 Name

A Returns the name of this object in the Collection of Alarm's User Fields. This value can be used as a parameter for that Collection's **Item** method.

5.2.3.1.4 Value

v Returns or configures the current value of this Alarm's User Field. This property has a different behavior depending whether this object is active or inactive. With this object active, the value returned when reading this property obeys the following priorities:

1. If there is a forced value (the **ValueSource** property equal to **evsForcedValue**), returns this value
2. If there is a configured Link (the **ValueSource** property equal to **evsLink**), returns the Link's current value
3. Searches for the value of this User Field in the hierarchically superior Area (traverses the hierarchy of Areas in ascending order)
4. If there is no hierarchically superior Area with a forced value or a configured Link for this User Field, retrieves the default value of the User Field configured in the Alarm Server

If this object is inactive, reading this property returns a forced value, if it exists (the **ValueSource** property equal to **evsForcedValue**). If there is no forced value, reading this property fails.

The behavior of writings to this property is the same, whether it is active or inactive. That writing fails if there is a configured Link (the **ValueSource** property equal to **evsLink**). Otherwise, the new value is accepted and the **ValueSource** property is automatically configured to **evsForcedValue**.

5.2.3.1.5 ValueSource

Specifies the source of the **Value** property of this Alarm's User Field. Possible values for this property are the following:

- **0 - evsInherited**: The **Value** property is inherited from the Alarm Server or from the parent Area (default value)
- **1 - evsLink**: The **Value** property is provided by the **Link** property
- **2 - evsForcedValue**: The **Value** property is a user-provided value

This property accepts writings with this object active, as well as inactive. In both cases, it is not possible to write the **evsLink** value. To change the **ValueSource** property to **evsLink**, users must write directly to the **Link** property (deactivating this

object if at run time).

NOTE: When exporting an Alarm's User Field object, the **ValueSource** property must be placed after the **Link** and **Value** properties in the order of columns.

5.3 Server's Runtime Objects

This section contains information about object available only at run time, the Server (Application) and the Application Folders.

5.3.1 Server

This section contains information about specific **methods** of the Server (Application) object. This object has the **Item** and **Save** general methods, besides the **Name** and **Count** properties, which are described in the **General Events, Methods, and Properties of Objects**. This object does not have events associated to it.

5.3.1.1 Methods

This section contains information about specific methods of the Server object.

NOTE: Methods described here can only be used at run time, they are not available in Studio.

5.3.1.1.1 ClearFailure

ClearFailure(*FailureName*)

This method is called to indicate that a failure reported by **ReportFailure** becomes inactive. The *FailureName* parameter specifies a failure name (user-defined) and must be passed to the **ReportFailure** method when that method is called.

5.3.1.1.2 E3GetActor

E3GetActor()

This method returns the logon name of the user who started the current request in E3Run. If E3Run is not currently handling an operation generated by another process, returns "System". If there is no user logged on the process which generated the current request, returns "Anonymous".

5.3.1.1.3 LoadReport

LoadReport(*ReportName*)

Loads a Report model. See Viewer's **LoadReport** method for an example of this

method's usage.

NOTE: The Report's **PrintPreview** method is not available in Reports loaded using this method.

5.3.1.1.4 ReportFailure

ReportFailure(*FailureName*, *FailureDescription*, *FailureWeight*)

Allows the application to report failures to the Server. The reported failures are usually conditions that keep the application from working partially, such as errors or malfunctions, and normally cannot be detected by the Server. Thus, this method can be used in these situations:

- To advise the system operator about problems in the server
- To help Hot-Standby manager to elect the best server to take control of the application

NOTE: The **ReportFailure** method is only available in the Server.

The **ReportFailure** method has the following parameters:

ReportFailure method parameters

NAME	DESCRIPTION
FailureName	Failure name (user-defined). This parameter must be passed to **ClearFailure** method whenever the failure becomes inactive. For example, "FaultCOM1".
FailureDescription	Failure description (user-defined). For example, "Communication failure at COM1".
FailureWeight	Defines failure's severity (or importance). A zero value indicates no gravity. Higher values indicate more severe faults.

Example:

```
Sub TagSerialStatus_OnValueChanged()
  If Value Then
    ' The tag value is True, application is on error
    Application.ReportFailure "FAULT_COM1", _
      "Communication fault at COM1", 100
  Else
    ' The tag value is False, clears the failure
    Application.ClearFailure "FAULT_COM1"
  End If
```

```
End Sub
```

5.3.1.1.5 Trace

Trace(*MessageText*[, *LogTimeStamp*[, *BreakLine*]])

Allows recording messages in a text file. Messages will be recorded in a file with the same name and path as the Domain file, but as a .txt file. Each new message is always inserted at the end of the file. In case there is some failure in file recording (like access denied, for example), there will be a script error.

This method can be used, for example, to record script debugging messages not executed at Viewer (because in this case it is not possible to use **MsgBox** method).

Trace method parameters

NAME	DESCRIPTION
MessageText	User-defined text message.
LogTimeStamp	(Optional) a Boolean indicating whether each record must have date and time (timestamp). If omitted, assumes True.
BreakLine	(Optional) a Boolean indicating whether there is a line break at the end of each message. If False, all records in the file are in a single line. If omitted, assumes True.

5.3.1.1.6 TrackEvent

TrackEvent(*EventMessage*, *Comment*, *TimeStamp*)

This method allows manually generating events via script. See Viewer's **TrackEvent** event for more information.

5.3.2 Application Folder

The Application Folder object is similar to **Data Folder** object, from **Data Server**, allowing to group Server objects inside folders. However, there are important differences:

- The Application Folder object is only available at run time
- This object has **Application**, **Count**, **Name**, **Parent**, and **PathName** general properties, and also **Item** and **Save** general methods
- This object allows using **For Each** VBScript command to enumerate its child objects, but that kind of access only takes into account Server objects, not listing, for example, screens and resources inside folders
- The root folder from where it is possible to enumerate objects is always the **Server** object

5.4 Configuration-Only Objects

This section contains information about objects which can be only used at configuration time.

5.4.1 E3StudioApplication

This section contains information about specific **methods** of the E3StudioApplication object. This object does not have specific events nor properties.

5.4.1.1 Methods

This section contains information about specific methods of the E3StudioApplication object.

NOTE: Methods described here can be only used in Studio, they are not available at run time.

5.4.1.1.1 CreateFile

CreateFile(*ProjectName*, *ClassName*[, *FileName*, *FolderName*, *RunWizard*, *OpenView*])

Creates a new object on a project file (.prj). Parameters for this method are the following:

- **ProjectName**: the name of the project file where the object will be inserted. This parameter cannot be empty, and the project file must be loaded on Studio. It can be project file's full path, or a Domain file's relative path. There is no need to add the .prj extension
- **ClassName**: The name of the object's class to be created. See the table below for possible values for this parameter
- **FileName**: The name of the inserted object. If this parameter is omitted, then it will be used a default name to create an object of the type indicated by *ClassName*. The maximum allowed size for this parameter is 32 characters, and if it already exists, it is automatically incremented
- **FolderName**: The name of the Folder where the object will be created. If this parameter is omitted, the object is then created at project file's root. The Folder name must obey the same rules of **Note**'s section of the **RenameFolder** method
- **RunWizard**: A **Boolean** indicating whether the configuration wizard of the class indicated by *ClassName* will be opened after executing this method. The default value of this parameter is True
- **OpenView**: A **Boolean** indicating whether the editor of the inserted object will be

opened after executing this method. The default value of this parameter is True

Possible values for ClassName parameter

OBJECT TO CREATE	PARAMETER VALUE
Alarm Configuration	DB.AlarmConfig
Alarm Server	DB.AlarmServer
Database	DB.DBServer
Data Server	Panel.DataServer
Formula	DB.Formula
FrameSet	Panel.FrameSet
Historic	DB.Hist
I/O Driver	IODrv.IODriver
OPC Driver	IODrv.OPCDriver
Report	Panel.Report
Screen	Panel.Screen
Storage	DB.Historian
Viewer	Panel.Viewer

5.4.1.1.2 CreateFolder

CreateFolder(*ProjectName*, *FolderName*[, *ParentFolder*])

Creates a new Folder with the name defined by *FolderName*, inside project file (.prj) *ProjectName*, which parent object is indicated by *ParentFolder*. The following restrictions apply to this method:

- The project file indicated in parameter *ProjectName* must exist, and it must be opened on Studio
- If parameter *FolderName* is empty, then a default name will be used to create the Folder. For example, `Folder1`
- The maximum allowed size for parameter *FolderName* is 32 characters
- If the name indicated in parameter *FolderName* already exists, then it will be automatically incremented
- The name of the Folder in *FolderName* must obey the same rules of **Note**'s section of the **RenameFolder** method
- The Folder indicated in parameter *ParentFolder* must exist. If this parameter is omitted, the new Folder is then created at project file's root

5.4.1.1.3 CreatePRJ

CreatePRJ(*Filename*)

Creates a new project file (.prj) named *Filename*, and then adds it to the current

Domain, if it exists. If the parameter *Filename* is not an absolute path, then it will use a path relative to the Domain, or the default path for project files (saved on Windows Registry). The .prj extension is added automatically, if needed.

5.4.1.1.4 RenameFolder

RenameFolder(*OldName*, *NewName*)

Allows renaming a Folder in an open project file in Studio, whether it belongs to the Domain or not. The *OldName* parameter is the full path to the Folder to rename, and the *NewName* parameter is the new name, without a path.

NOTES:

- If Folder's full path in the parameter *OldName*, or the new name in the parameter *NewName* starts with a character that is not a letter (with no diacritical marks or "ç"), or if it has any character that is not a letter (with no diacritical marks or "ç"), a number (0-9), or an underscore, you must surround that name with brackets.
- If the parameter *NewName* is invalid, or if the parameter *OldName* has a syntax error, or else the path indicated by *OldName* is not found on any open project file, a script error will occur
- Password-protected project files must be unlocked before using this method

5.5 Drivers

This section contains information about events, methods and properties of the following objects: **I/O Driver**, **I/O Tag**, **I/O Block**, **I/O Block Element**, **OPC Driver**, **OPC Tag**, **OPC Block**, **OPC Block Element**, and **OPC UA Driver**.

5.5.1 I/O Driver

This section contains information about **events**, **methods** and **properties** of the I/O Driver object.

5.5.1.1 Events

This section contains information about the events of the I/O Driver object.

5.5.1.1.1 AfterStart

AfterStart()

Occurs after the I/O Driver started to communicate. It is common to create a script for this event using the **Write** method to perform device configurations. Example:

```
Sub Driver1_AfterStart()
  ' After started, communication writes values
  ' to the equipment or device
  Write 0, 2, 55, 2, 33.4
  Write 0, 3, 55, 20, "Metal"
```

```
End Sub
```

5.5.1.1.2 AfterStop

AfterStop()

Occurs after the I/O Driver finished the communication. Use the **AfterStop** event to perform any necessary actions after Driver communication finished.

5.5.1.1.3 BeforeStart

BeforeStart()

Occurs when the Driver is about to start the communication. Use the **BeforeStart** event to perform any necessary actions before starting communication, such as setting up Driver parameters. Example:

```
Sub Driver1_BeforeStart()
  ' It performs the startup of driver parameters before
  ' starting the communication
  P1 = 0
  P2 = 20
  P3 = 80
  P4 = 0
End Sub
```

5.5.1.1.4 BeforeStop

BeforeStop()

Occurs when the Driver is about to finish the communication. Use the **BeforeStop** event to perform any necessary actions before finishing communication, such as writing to or reading values from the equipment or device, before communication is no longer available.

5.5.1.1.5 OnCommError

OnCommError(*EvtType*, *Size*, *Element*, *N1*, *N2*, *N3*, *N4*)

Occurs when an I/O Driver detects any writing or reading error. Use an **OnCommError** event to check when a writing or reading failure occurred in the I/O Driver. Event variables receive information about this error. With these values, it is possible to track back Tags with I/O problems.

OnCommError event variables

NAME	DESCRIPTION
EvtType	Informs which type of operation the Driver was performing when the error occurred, according to the following options: • **0**: Single Element reading error (**Size** = 1). **Param1** is *N1*, **Param2** is *N2*, **Param3** is *N3*, and **Param4** is *N4* • **1**: Single Element writing error (**Size** = 1). **Param1** is *N1*, **Param2** is *N2*, **Param3** is *N3*, and **Param4** is *N4* • **2**: Block reading error (I/O Block). **Size** is determined by the number of Block Elements. **Param1** is *N1*, **Param2** is *N2*, **Param3** is *N3*, and **Param4** is *N4* • **3**: Block writing error (I/O Block). **Size** is determined by the number of Block Elements. **Param1** is *N1*, **Param2** is *N2*, **Param3** is *N3*, and **Param4** is *N4*
Size	Number of values being read or written.
Element	Index of the Element being read or written inside a Block.
N1	First parameter of a reading or writing operation that generated the error.
N2	Second parameter of a reading or writing operation that generated the error.
N3	Third parameter of a reading or writing operation that generated the error.
N4	Fourth parameter of a reading or writing operation that generated the error.

Example:

```
Sub Driver1_OnCommError(Type, Size, Element, N1, N2, N3, N4)
  Application.GetObject("Data.InternalTag1").Value = _
    Application.GetObject("Data.InternalTag1").Value + 1
  Application.GetObject("Data.EvtType").Value = EvtType
  Application.GetObject("Data.Size").Value = Size
  Application.GetObject("Data.Element").Value = Element
  Application.GetObject("Data.N1").Value = N1
  Application.GetObject("Data.N2").Value = N2
  Application.GetObject("Data.N3").Value = N3
  Application.GetObject("Data.N4").Value = N4
End Sub
```

5.5.1.1.6 OnCommErrorEx

OnCommErrorEx(*ErrorInfo*)

Occurs soon after the execution of the **OnCommError** event.

ErrorInfo parameter information

NAME	DESCRIPTION
ErrorInfo.EvtType	Indicates the type of operation that caused the error: • **0**: Tag reading • **1**: Tag writing • **2**: Block reading • **3**: Block writing
ErrorInfo.Size	Size of the Block that caused the error (if it is a Tag, **Size** is equal to 1).
ErrorInfo.Element	Index of the Block Element that caused the error.
ErrorInfo.Nx	N*x* or B*x* parameters (*x* = 1, 2, 3, 4) of the operation that caused the error.
ErrorInfo.ParamDevice	**ParamDevice** parameter (**String**) of the operation that caused the error.
ErrorInfo.ParamItem	**ParamItem** parameter (**String**) of the operation that caused the error.

5.5.1.1.7 OnTagRead

OnTagRead(*Tag*)

Occurs on Tag reading, whenever the I/O Driver returns a new value or an error. This means that if Tag value or Tag quality do not change, the event is not triggered. For this event to work, the **EnableDriverEvent** property must necessarily be enabled. In addition, the **PercentDeadband** property can also influence the event, if the **EnableDeadband** property is enabled. Example:

```
Sub Tags_OnTagRead(Tag)
  Set Obj = Application.GetObject("Data1.TagName")
  Obj.Value = Tag.Name
  Set Obj = Application.GetObject("Data1.TagRead")
  Obj.Value = True
  Set Obj = Application.GetObject("Data1.TagType")
  Obj.Value = TypeName(Tag)
End Sub
```

5.5.1.1.8 OnTagWrite

OnTagWrite(*Tag*, *Succeeded*, *User*)

Occurs when a writing operation is triggered on any Driver Tag.

OnTagWrite event variables

NAME	DESCRIPTION
Tag	A reference to the Tag object being written. For example, users can access the Tag property using a syntax such as **Tag.DocString**.
Succeeded	A **Boolean** value that indicates whether a writing operation was successful or not.
User	Parameter that receives the name of the user performing the writing operation. The default value of this parameter is "System". If there is no user logged in the Viewer generating this event, this parameter contains the value "Anonymous".

5.5.1.2 Methods

This section contains information about the methods of the I/O Driver object.

5.5.1.2.1 Write

Write(*N1*, *N2*, *N3*, *N4*, *Value*[, *WriteSyncMode*])

Writes data to the device. This method returns a **Boolean** indicating whether the operation was successful or not. The *N1* to *N4* parameters correspond to Driver's **N** parameters. The *Value* parameter defines the value to write to the Driver. The *WriteSyncMode* parameter allows users to use a writing mode other than the one used in the Driver. The available options for this parameter are the following:

- **0**: Uses the writing mode configured in the Driver (default)
- **1**: Executes a synchronous writing
- **2**: Executes an asynchronous writing (no confirmation)

If the *WriteSyncMode* parameter is omitted, the writing mode configured in the Driver is also used. For more information about these parameters, please check the Driver's documentation.

5.5.1.2.2 WriteEx

WriteEx(*N1*, *N2*, *N3*, *N4*, *Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes data to the equipment. This method returns a **Boolean** indicating whether the operation was successful or not. *N1* to *N4* parameters correspond to Driver's *[N]* parameters. The *Value* parameter defines the value to write to the Driver. For more information on these parameters, please check the Driver's documentation.

The *Timestamp*, *Quality*, and *WriteStatus* parameters are optional. When omitted, this method's behavior is similar to the **Write** method. The *Timestamp* parameter specifies the time stamp to write to the Tag, if supported by the equipment. If omitted, assumes the time stamp from the moment of the writing operation. The *Quality* parameter indicates quality, from 0 to 255. If omitted, assumes a Good (192) quality. The *WriteStatus* parameter receives a value returned by the Driver, indicating the writing status (if supported by the Driver, according to its own documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. Example:

```
Dim status
If tag001.WriteEx(0, 0, 0, 0, 100, , , status) Then
  MsgBox "Successful writing, status = " & status
Else
  MsgBox "Writing failed, status = " & status
End If
```

5.5.1.3 Properties

This section contains information about the properties of the I/O Driver object.

5.5.1.3.1 DisableIOServerPool

◩ Forces the Driver to not participate in the Pool of IOServer processes, that is, the Driver runs in an exclusive IOServer. The default value of this property is False.

NOTE: For more information, please check the topic **Pool of IOServer Processes**, in the **User's Manual**.

5.5.1.3.2 DriverLocation

A Defines what Driver is used by the I/O Driver object to communicate with a device. This property accepts a **String** with the complete Driver's path, in case it is not in the same Domain's directory, or the relative Driver's path if it is in the same Domain's directory. In case there is no open Domain on Studio, a relative path is considered as starting from the folder where the project or library containing the Driver object is. After that, the **DriverName** property changes to the Driver's description. This

property cannot be modified once communication has started. Default value is an empty **String**.

NOTE: It is always advisable to use the option **Browse DLL**, on I/O Driver object's contextual menu, to set this property correctly.

5.5.1.3.3 DriverName

A Contains a **String** describing the Driver associated to the I/O Driver object. For this, users must first set the **DriverLocation** property. This is a read-only property, and it is not available at run time.

5.5.1.3.4 EnableReadGrouping

◩ Enables reading optimization (automatic Tag grouping). This property cannot be changed at run time. Default value is True. The reading optimization only happens if the Driver supports this feature.

5.5.1.3.5 P1

9 Use the **P1** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.6 P2

9 Use the **P2** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.7 P3

9 Use the **P3** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
End Sub
```

5.5.1.3.8 P4

9 Use the **P4** property to configure the Driver. Check the Driver documentation to correctly set its parameters. This property cannot be modified after communication has started. Example:

```
Sub Driver1_BeforeStart()
  ' Driver1 is an object of type I/O Driver
  DriverLocation = "c:\driver\plc.dll"
  P1 = 2
  P2 = 1
  P3 = 9600
  P4 = 500
End Sub
```

5.5.1.3.9 ParamDevice

A Defines the address of the device accessed by the Driver. This property is inherited by the Driver's Blocks and Tags, which can override this value, if necessary.

5.5.1.3.10 ReadRetries

9 Indicates the number of Driver's reading retries, in case of error. If it is set to 2, for example, it means the Driver retries failed communications twice, apart from the original try.

5.5.1.3.11 ShareMaximum

9 Defines the maximum number of I/O Drivers that is grouped into a shared I/O Server. This property is only used when the **ShareServer** property is enabled. Example:

```
' This driver will not be shared
ShareServer = False
ShareMaximum = <any value>
```

```
' All drivers are grouped into the same IOServer
' Does not define a limit
ShareServer = True
ShareMaximum = 0
' Groups every 5 drivers into an IOServer
ShareServer = True
ShareMaximum = 5
```

5.5.1.3.12 ShareServer

◩ If True, this Driver shares its execution among all other I/O Drivers that have the same **String** in the **DriverLocation** property. It means that only the first I/O Driver that is set executes the communication initialization procedure. All other shared I/O Drivers ignore all setting parameters from *P1* to *P4*, as well as other settings. Otherwise, if the **ShareServer** property is False, the Driver does not share any kind of communication with other I/O Drivers. This property cannot be modified once communication has started. Default value is False.

5.5.1.3.13 WriteFeedbackMode

◩ Allows controlling writing feedback confirmation to Tags. It applies only to readable Tags, that is, Tags whose **AllowRead** property is set to True. Through this property, reading Tags that receive writings is quickier. This property has the configuration options described on the next table.

Available options for the WriteFeedbackMode property

OPTION	DESCRIPTION
0 - wfWaitNextRead	Tag reading is performed normally on the next scan.
1 - wfImmediateReadAfterWrite	After each writing, a confirmation reading is performed as soon as possible.
2 - wfTrustWriteSuccess	If the Driver indicates a successful writing, the written value is directly assumed by the Tag, without reading it from the PLC.

The default value is **1 - wfImmediateReadAfterWrite**. Applications created on earlier versions, before this property existed, assume 0 (zero) when loaded. Example:

```
Sub CommandButton1_Click()
  Dim mode
  mode = Application.GetObject("Driver1").WriteFeedbackMode
  MsgBox mode
  Select case mode
  Case 0
    MsgBox "Tag reading will be done in the next scan."
  Case 1
```

```
    MsgBox "After each writing, a confirmation _
      reading will be done as soon as possible."
  Case 2
    MsgBox "If the driver indicates writing success, _
      the written value is assumed directly by the tag, _
      without reading it from the PLC."
  End Select
End Sub
```

NOTE: When a value of 2 is used, time stamp and quality may be incorrect, since in a successful writing the value is assumed by the Tag without searching for time stamp and quality in the PLC. In addition, the assumed value itself may present a little deviation due to any kind of simplification in numbers that may occur in the Driver or in the PLC. Also, note that some Drivers or protocols may indicate success, even when the writing has failed. So, the other values (**0 - wfWaitNextRead** or **1 - wfImmediateReadAfterWrite**) should be preferred when possible.

5.5.1.3.14 WriteRetries

9 Indicates the number of Driver's writing retries, in case of error. If it is set to 2, for example, it means the Driver retries a failed communication twice, apart from the original try.

5.5.1.3.15 WriteSyncMode

Determines how writings are sent to the I/O Server (synchronous or asynchronous mode). This property has the configuration options described on the next table.

Available options for the WriteSyncMode property

OPTION	DESCRIPTION
0 - wsmDefault	Synchronous mode (default).
1 - wsmSync	Synchronous mode. Every time a value is written on a Tag, E3Run sends the writing to the I/O Server and waits for the return of the writing.

OPTION	DESCRIPTION
2 - wsmAsyncUnconfirmed	Asynchronous mode without confirmation. All writings are sent to the I/O Server without waiting its return, and it is always assumed that the writing was successful. When in asynchronous mode, the Tag's writing methods (**Write**, **WriteEx**) always return True immediately, and the writing status (on methods returning this status) remains always empty. The Driver's **OnTagWrite** event is executed as soon as the writing is sent to the I/O Server, and the *Succeeded* parameter always remain in True.

Asynchronous writings are executed by the I/O Server as soon as the Driver is available (when current reading is over). If several asynchronous writings are sent to the I/O Server, the Driver only returns the readings after all asynchronous writings are executed.

5.5.1.4 I/O Block

This section contains information about **events**, **methods**, and **properties** of the I/O Block object.

5.5.1.4.1 Events

This section contains information about the events of the I/O Block object.

5.5.1.4.1.1 OnRead

OnRead()

Occurs when the Driver performs an I/O Block reading. Use the **OnRead** event when it is necessary to perform some operation soon after some data has been modified in the I/O Block object, such as the **Quality**, **TimeStamp** or even the **Value** property of some of the Block's Element. Example:

```
Sub IOBlock1_OnRead()
  ' When reading a block, assign to the InternalTag1 tag
  ' the value of the block element elm1.
  Set obj=Application.GetObject("DataServer1.InternalTag1")
  Set elm = Application.GetObject("Driver1.IOBlock1.elm1")
  obj.Value = elm.Value
End Sub
```

5.5.1.4.2 Methods

This section contains information about the methods of the I/O Block object.

5.5.1.4.2.1 Write

Write([*WriteSyncMode*])

Writes the I/O Block's current value in the equipment. Usually, this script command is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. For more information, please check the Driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

5.5.1.4.2.2 WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All parameters are optional and, if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to write to the Driver. Data type depends on the Driver and, if omitted, Tag's current value is assumed. The *Timestamp* parameter specifies the time stamp to write to the Tag, if supported by the equipment. If omitted, assumes the time stamp from the moment of the writing operation. The *Quality* parameter indicates quality, from 0 to 255. If omitted, assumes a Good (192) quality. The *WriteStatus* parameter receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode

- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.4.3 Properties

This section contains information about the properties of the I/O Block object.

5.5.1.4.3.1 AdviseType

Controls **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is updated if the **AllowRead** property is True.
1 - AdviseWhenLinked	The Tag is updated only if the **AllowRead** property is True, and the Tag is associated to an active object, that is, one Display in an open Screen or an enabled Alarm, among others. A Tag link for this purpose can be assigned to the **Value**, **RawValue**, **Quality**, and from **Bit00 to Bit31** for Block Element properties, and **Quality** and **TimeStamp** for I/O Block properties.

5.5.1.4.3.2 AllowRead

Defines whether the Block is read by the I/O Driver. If True, the Driver automatically updates the I/O Elements inserted in the Block, in time spans defined by the **Scan** property. Otherwise, the I/O Block is neither read nor updated. This property can be modified at run time. Default value is True. Example:

```
Sub Button1_Click()
  'Stops block reading
  Set obj = Application.GetObject("Driver1.block1")
  obj.AllowRead = False
End Sub
```

5.5.1.4.3.3 AllowWrite

◩ Defines whether the Block is written automatically when the **Value** property from its I/O Block Elements is modified. If True, the modifications are sent to the device associated to the I/O Driver. Otherwise, the modifications are ignored. I/O Block Elements do not accept values if this property is set to False, unless the **AllowRead** property is also set to False. Example:

```
Sub Button1_Click()
  'Disables block writing
  Set obj = Application.GetObject("Driver1.block1")
  obj.AllowWrite = False
End Sub
```

5.5.1.4.3.4 B1

9 Specifies to which data set in the device this Tag is linked. Please check the driver's documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.5 B2

9 Specifies to which data set in the device this Tag is linked. Please check the driver's documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.6 B3

9 Specifies to which data set in the device this Tag is linked. Please check the driver's documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
End Sub
```

5.5.1.4.3.7 B4

9 Specifies to which data set in the device this Tag is linked. Please check the driver's documentation for appropriate parameters. This property can be modified once communication has started. Example:

```
Sub Block1_BeforeStart()
  B1 = 2
  B2 = 1
  B3 = 9600
  B4 = 524
End Sub
```

5.5.1.4.3.8 EnableDeadBand

◩ Enables or disables the **PercentDeadBand** property of Block Elements. If True, the Block value is only updated when it changes, and its new value exceeds the limit defined by the **PercentDeadBand** property of any Block Element. Otherwise, the Block is always updated, and the dead band limit is not checked. Whenever possible, users should keep the dead band enabled, because it enhances data acquisition and the processing performance. Usually, dead band is disabled only for Tags that return values representing events that need to be handled on the **OnRead** event. The default value of this property is True.

NOTES:

- In case there are more than one Block Element mapped to the same index, the dead band configuration used is the one that results in the smallest absolute dead band value
- If any Block index has an unmapped Element, the dead band in this index is equal to 0, that is, any variation in the value of the Element violates the dead band
- If the **EnableDeadBand** property is enabled, the last value sent to the Block is compared to the current value read, Element by Element. If any Block Element violates its dead band, the whole Block is updated

5.5.1.4.3.9 EnableDriverEvent

◩ Controls the way **OnTagRead** events are generated, which occur in the I/O Driver that contains the Block. If True, the **OnTagRead** event generation is then enabled by this Tag. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tags, I/O Blocks, and Block Elements) can generate this event. The event occurs in the Driver, not in the Block.

5.5.1.4.3.10 ParamDevice

A Defines the address of the device accessed by the Block. This property is inherited from the Driver, but its value can be overridden, if necessary.

5.5.1.4.3.11 ParamItem

A Identifies data accessed by the Block from the device.

5.5.1.4.3.12 Quality

9 Informs the quality of the value contained in the **Value** property. Each time the Driver sets a new value to the Block, it also sets data quality. This is a read-only property. Default value is 0 (Bad Quality).

NOTE: For more information about quality, please check the topic **Quality** on **E3 User's Manual**.

5.5.1.4.3.13 Scan

9 Specifies the duration of the scan used by the server to update the Block. This property is represented in milliseconds, and can be modified after communication has started. It is used only when the **AllowRead** property is set to True. When users set this property in several Blocks in the application, they should increase the value of this property for those Blocks that do not vary very much in the device, enabling other Blocks with higher priority to be read more frequently, thus enhancing general application performance. Default value is 1,000 (1 second). Scan value must be greater than zero. Example:

```
Sub IOBlock1_BeforeStart()
  Scan = 152
End Sub
```

5.5.1.4.3.14 Size

9 Defines the size of value sets in this Block. Please check the driver's documentation for this property's limits, according to *B1* to *B4* parameters. By creating Elements for the Block, users can access reading and writing values for the device. This property cannot be modified once communication has been started. Default value is 0 (zero). Example:

```
Sub IOBlock1_BeforeStart()
  Size = 10
End Sub
```

5.5.1.4.3.15 TimeStamp

Updated whenever there is a change in value or status in either the **Value** or the **Quality** properties. Informs time stamp associated to both value and quality in an I/O Block. This is a read-only property. Default value is 00:00:00.

5.5.1.4.4 I/O Block Element

This section contains information about **methods** and **properties** of the I/O Block Element object. This object does not have events associated to it.

5.5.1.4.4.1 Methods

This section contains information about the methods of the I/O Block Element object.

Write

Write([*WriteSyncMode*])

Writes I/O Block Element's current value to the equipment. Usually, this script is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. For more information, please check the driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All its parameters are optional and, if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to write to the Driver. Data type depends on the Driver and, if omitted, Tag's current value is assumed. The *Timestamp* specifies the time stamp to write to the Tag, if supported by the equipment. If omitted, assumes the time stamp from the moment of the writing operation. The *Quality* indicates quality, from 0 to 255. If omitted, assumes a Good (192) quality. The *WriteStatus* receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its own documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.4.4.2 Properties

This section contains information about the properties of the I/O Block Element object.

Bit00 to Bit31

◩ These properties represent the 32 bits in I/O Tag's **Value** property, ranging from **Bit00** (the least significant bit) to **Bit31** (the most significant bit). By modifying each one of these bits, users modify Tag's **Value** property, and vice versa, but this only happens when the **UseBitFields** property is True. Default value is False.

NOTE: Bit values (from **Bit00** to **Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

DeviceHigh

9 Defines the highest value reached by the Block Element in the device. This property is used in the calculation of the **PercentDeadBand** property, and also to adjust the device's scale value before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is set to True. Default value is 1000.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by the scale adjustment, that is, they represent bits from the value read by the equipment before conversion.

DeviceLow

9 Defines the lowest value reached by the Block Element in the device. This property is used in the calculation of the **PercentDeadBand** property, and also to adjust the device's scale value before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This

conversion happens only when the **EnableScaling** property is set to True. Default value is 0.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by the scale adjustment, that is, they represent bits from the value read by the equipment before conversion.

EnableDriverEvent

◩ Controls how the **OnTagRead** event is generated, which occurs in the I/O Driver that contains the Block. If this Tag's **EnableDriverEvent** property is configured to True, generating the **OnTagRead** event is enabled by this Tag. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tag, I/O Block, and Block Element) can generate this event. The event occurs in the Driver, and not in the Block.

EnableScaling

◩ If True, all values from the device receive scale adjustments according to the **DeviceHigh**, **DeviceLow**, **EUHigh** and **EULow** properties before they are attributed to the **Value** property. Otherwise, no adjustment in scale are performed. Default value is False. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

9 Defines the highest value to be attributed to the **Value** property, adjusting the scale to the value from the device before it is done. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. The default value of this property is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

EULow

9 Defines the lowest value to be attributed to the **Value** property, adjusting the scale to the value from the device before it is performed. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when the **EnableScaling** property is True. The default value of this property is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

Index

9 Specifies this Element's position among the other Elements configured by the I/O

Block's **Size** property where it is inserted. This property's value range from 0 to the value defined by **Size** minus 1. (For example: If **Size** is 20, **Index** ranges from 0 to 19). This property can be modified once communication has started. Default value is 0, but when mapping the Elements from a Block, Studio automatically adjusts **Index** parameter to the specified value. Example:

```
Sub Element_OnStartRunning()
  Index = 15
End Sub
```

PercentDeadBand

9 The **PercentDeadBand** property determines the minimum variation (dead band) of a Block Element, so that this value can be updated in E3. This value is specified as a percentage of the difference between the **DeviceHigh** and the **DeviceLow** properties. This property is only used if the **EnableDeadBand** property of the Block is set to True. If the **PercentDeadBand** property is equal to 0, the Block Element does not have a dead band, and any variation in its value is passed to E3. Otherwise, E3 receives a new value only if its difference, relative to the current value, is larger than the dead band. The default value of this property is 0.

Quality

9 Informs the quality of the value contained in the **Value** property. Each time the Driver attributes a new value to the Tag, it also sets data quality. This property is read-only. Default value is 0 (Bad Quality).

NOTE: For more information on quality, please check the topic **Quality** on **E3 User's Manual**.

RawValue

v Accesses the Element's original value previous to the scale, that is, regardless of **EnableScaling** property configuration. Therefore, if this property is set to False, the **Value** and **RawValue** properties behave identically.

UseBitFields

◩ If this property is set to True, every time the value from **Value** property changes, the bits referring to **Bit00 to Bit31** properties are updated. Also, every time the value from **Bit00 to Bit31** properties are modified, the value from **Value** property is updated and sent to the device, if the **AllowWrite** property is True. Otherwise, bits do not lead to any modification. This property can be updated once communication has started. Default value is False.

Value

v Updated whenever a new valid reading from a device is performed, using *N1* to *N4* parameters. Data type (**Integer**, **Floating Point**, or **String**) depends on the Driver to which the Element is associated and its parameters.

This property is only updated this way if the **AllowRead** property is True, and when

there are no communication errors (in this case, the **Quality** and **TimeStamp** properties are updated), according to a scan time defined in the **Scan** property. Another way to use this property is to write values in the device, just by setting a new value to the **Value** property or to some of the bits from **Bit00 to Bit31**. In this case, the **AllowWrite** property must be True.

This is also I/O Tag's default property. Therefore, a reference by value to a Block Element does not need to explicit the **Value** property to access this value. If this property is not being updated, check if the **Index** property is correctly configured. Default value is empty (no value). Example:

```
Sub Button1_Click()
  ' Accesses a tag and shows current value
  ' tag1 is an I/O tag
  Set obj = Application.GetObject("IODriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  'This can also be done in a different way,
  'with no need to show Value property, which is default
  MsgBox " Tag1's current value: " & obj
End Sub
```

5.5.1.5 I/O Folder

This section contains information about **properties** of the I/O Folder object of the I/O Driver. This object does not have events nor methods associated to it.

5.5.1.5.1 Properties

This section contains information about the properties of the I/O Driver's I/O Folder object.

5.5.1.5.1.1 ParamDevice

A Defines the address of the device accessed by the I/O Folder. This property is inherited from the I/O Driver, and its value can be overwritten, if needed.

5.5.1.6 I/O Tag

This section contains information about **events**, **methods** and **properties** of the I/O Tag object.

5.5.1.6.1 Events

This section contains information about the events of the I/O Tag object.

5.5.1.6.1.1 OnRead

OnRead()

Occurs when a Tag reading is performed by the Driver. Use the **OnRead** event when it is necessary to perform some operation soon after some data has been modified in the Tag, such as the **Value**, **Quality**, or **TimeStamp** properties. This event is generated by a background reading. Example:

```
Sub CommTag1_OnRead()
  ' When reading the tag, assign its value
  ' to the InternalTag1 tag.
  Set obj = Application.GetObject("DataServer1.InternalTag1")
  obj = Value ' CommTag1 Value
End Sub
```

5.5.1.6.2 Methods

This section contains information about the methods of the I/O Tag object.

5.5.1.6.2.1 Write

Write([*WriteSyncMode*])

Writes I/O Tag's current value to the equipment. Usually, this method is used only when the object's **AllowWrite** property is False.

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. For more information, please check the driver's documentation. This method returns a **Boolean** indicating whether the operation has been successful or not.

5.5.1.6.2.2 WriteEx

WriteEx(*Value*, *Timestamp*, *Quality*[, *WriteStatus*[, *WriteSyncMode*]])

Writes values to the equipment. All parameters are optional and, if omitted, method's behavior is the same as the **Write** method. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *Value* parameter defines the value to write to the Driver. Data type depends on

the Driver and, if omitted, assumes Tag's current value. The *Timestamp* specifies the time stamp to write to the Tag, if supported by the equipment. If omitted, assumes the time stamp from the moment of the writing operation. The *Quality* parameter indicates quality, from 0 to 255. If omitted, assumes a Good (192) quality. The *WriteStatus* parameter receives a value returned by the Driver, indicating writing status (if supported by the Driver, according to its documentation).

The *WriteSyncMode* parameter allows users to use a writing mode that may be different from the one used in the Driver. The available options for this parameter are the following:

- **0**: Writing mode configured in the Driver
- **1**: Synchronous writing mode
- **2**: Asynchronous writing mode (no confirmation)

When this parameter is omitted, the writing mode configured in the Driver is also used. Example:

```
Sub Tag1_OnRead()
  ' Use WriteEx to transfer values from one driver to another
  Application.GetObject("Driver2.Tag")._
    WriteEx Value, TimeStamp, Quality
End Sub
```

5.5.1.6.3 Properties

This section contains information about the properties of the I/O Tag object.

5.5.1.6.3.1 AdviseType

Controls the **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is kept updated if the **AllowRead** property is equal to True.
1 - AdviseWhenLinked	The Tag is only updated if the **AllowRead** is equal to True and it is linked to any active object, for example a Display on an open Screen or an enabled alarm, among others. The Tag link for this purpose can be performed on the following properties: **Value**, **RawValue**, **TimeStamp**, **Quality**, and from I/O Tag's **Bit00 to Bit31**.

Example:

```
Sub CommandButton3_Click()
```

```
  MsgBox Application._
    Application.GetObject("Driver1.Tag1").AdviseType
End Sub
```

5.5.1.6.3.2 AllowRead

◩ Defines whether this Tag is read by the I/O Driver. If True, the Driver automatically updates **Value** and **Bits** (from **Bit00** to **Bit31**) properties, in time spans defined by the **Scan** property. Otherwise, the I/O Tag is neither read nor updated. This property can be modified at run time. Default value is True. Example:

```
Sub Button1_Click()
  ' Stops tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

5.5.1.6.3.3 AllowWrite

◩ Defines whether this Tag is written automatically when **Value** or **Bits** (from **Bit00** to **Bit31**) properties are modified. If True, the modifications are sent to the device associated to the I/O Driver. Otherwise, the modifications are ignored. Default value is True. Example:

```
Sub Button1_Click()
  ' Disables tag writing
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

5.5.1.6.3.4 Bit00 to Bit31

◩ These properties represent the 32 bits in I/O Tag's **Value** property, ranging from **Bit00** (the least significant bit) to **Bit31** (the most significant bit). By modifying each one of these bits implies in modifying Tag's **Value** property, and vice versa, but this only happens when the **UseBitFields** property is True. Default value is False.

5.5.1.6.3.5 DeviceHigh

9 Defines the highest value achieved by this Tag in the device. This property is used to perform a scale adjustment of the value received from the device before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion only occurs when the **EnableScaling** property is True. Default value is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
```

```
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.6 DeviceLow

9 Defines the lowest value achieved by this Tag in the device. This property is used to perform a scale adjustment of the value received from the device before setting it to the **Value** property. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion only occurs when the **EnableScaling** property is True. Default value is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.7 EnableDeadBand

◩ Enables or disables the **PercentDeadBand** property. If True, Tag value is only updated when it changes, and its new value exceeds the limit defined by the **PercentDeadBand** property. Otherwise, the Tag is always updated, and the dead band limit is not checked. Whenever it is possible, users should keep dead band enabled, because it enhances data acquisition and processing performance. Usually, users should disable dead band only for Tags that return values representing digital or analogical events, and it is necessary to process these events via script on Tag's **OnRead** event. Default value is True.

5.5.1.6.3.8 EnableDriverEvent

◩ Controls how the **OnTagRead** event is generated, which occurs in the I/O Driver that contains the Block. If this Tag's **EnableDriverEvent** property is configured to True, then this Tag's **OnTagRead** event is enabled. Otherwise, it is disabled. The three kinds of I/O Elements (I/O Tag, I/O Block, and Block Element) can generate this event. The event occurs in the Driver, and not in the Block.

5.5.1.6.3.9 EnableScaling

◩ Enables or disables a value scale for the value sent to or received from a device. If this property is set to True, all values from the device receive scale adjustments on **DeviceHigh**, **DeviceLow**, **EUHigh** and **EULow** properties before attributing it to the **Value** property. Otherwise, no adjustment in the scale are performed (reading or writing). Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

5.5.1.6.3.10 EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

5.5.1.6.3.11 EUHigh

9 Defines the highest value to set to the **Value** property, adjusting the scale to the value from the device before this attribution. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion

happens only when the **EnableScaling** property is True. The default value of this property is 1,000. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.12 EULow

Defines the lowest value to set to the **Value** property, adjusting the scale to the value from the device before this attribution. Likewise, the inverse operation is performed before sending the value to the Driver, when writing. This conversion happens only when **EnableScaling** property is True. The default value of this property is 0. Example:

```
Sub Tag_OnStartRunning()
  ' Adjusts the scale of temperature
  ' ranging from 0 to 255 at PLC, but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by the adjustment in scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.13 N1

Specifies the device's variable that this Tag is linked. Please check the driver's documentation for an appropriate configuration. This property can be modified once communication has started. Default value is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
```

```
End Sub
```

5.5.1.6.3.14 N2

9 Specifies the device's variable that this Tag is linked. Please check the driver's documentation for an appropriate configuration. This property can be modified once communication has started. Default value is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  N2 = 3
End Sub
```

5.5.1.6.3.15 N3

9 Specifies the device's variable that this Tag is linked. Please check the driver's documentation for an appropriate configuration. This property can be modified once communication has started. Default value is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
  N3 = 5
  N4 = 20
End Sub
```

5.5.1.6.3.16 N4

9 Specifies the device's variable that this Tag is linked. Please check the driver's documentation for an appropriate configuration. This property can be modified once communication has started. Default value is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  N1 = 10
  N4 = 20
End Sub
```

5.5.1.6.3.17 ParamDevice

A Defines the address of the device accessed by the Tag. This property is inherited from the Driver, but its value can be overwritten, if necessary.

5.5.1.6.3.18 ParamItem

A Identifies data this Tag accesses from inside the device.

5.5.1.6.3.19 PercentDeadBand

9 The **PercentDeadBand** property determines the minimum variation of a Tag's value (dead band), so that this value can be updated on E3. This value is specified as a percentage of the difference between the **DeviceHigh** and **DeviceLow** properties. This property is only used if the Tag's **EnableDeadBand** property is set to True. If the **PercentDeadBand** property is set to 0 (zero), the Tag has no dead band, and any

variation in its value is sent to E3. Otherwise, E3 only receives a new value whose difference, relative to E3's current value, is larger than dead band. Default value is 0 (zero).

5.5.1.6.3.20 Quality

The Quality property informs which is the quality of the value contained in the **Value** property. Every time the Driver attributes a new value to the Tag, it also configures the quality of that data. This is a read-only property. Default value is 0 (Bad Quality).

NOTE: For more information on quality, please check the topic **Quality** on **E3 User's Manual**.

5.5.1.6.3.21 RawValue

Accesses Tag's original value before the scale, that is, regardless of the configuration on the **EnableScaling** property. Therefore, if this property is False, the **Value** and **RawValue** properties behave identically.

5.5.1.6.3.22 Scan

Use the **Scan** property to specify a scan time used by the server to update the **Value** property. This property is represented in milliseconds and can be modified after starting communication, and it is used only when the **AllowRead** property is set to True. When configuring this property in several application Tags, users should increase the value of the **Scan** property to those Tags that do not vary very much in the device, enabling other Tags with higher priority to be read more frequently, thus enhancing general application performance. Default value is 1,000 (1 second). The scan value must be greater than 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  Scan = 1500
End Sub
```

5.5.1.6.3.23 TimeStamp

The **TimeStamp** property is updated whenever there is a change in value or status in either **Value** or **Quality** properties. It informs time stamp associated to both value and quality in an I/O Tag. This is a read-only property. Default value is 00:00:00.

5.5.1.6.3.24 UseBitFields

If the **UseBitFields** property is True, every time the value from **Value** property is modified, it updates the bits referring to the **Bit00 to Bit31** properties. Likewise, every time the value from **Bit00 to Bit31** properties are modified, the value from **Value** property is updated and sent to the device, if the **AllowWrite** property is True.

Otherwise, bits do not lead to any modification. This property can be updated once communication has started.

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by adjusting the scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.1.6.3.25 Value

Updated whenever a new valid reading of a device is performed, using *N1* to *N4* parameters. Data type (**Integer**, **Floating Point**, or **String**) depends on the Driver to which the Element is associated and its parameters.

This property is updated this way if the **AllowRead** property is True, and when there are no communication errors (in this case, the **Quality** and **TimeStamp** properties are updated), according to a scan time defined in the **Scan** property. Another way to use this property is to write values to a device, by simply setting a new value to the **Value** property or to some of the bits from the **Bit00 to Bit31** properties. In this case, the **AllowWrite** property must be True.

This is also I/O Tag's default property. Therefore, a reference by value to an I/O Tag does not need to explicit the **Value** property to access its value. Default value is empty (no value). Example:

```
Sub Button1_Click()
  ' Accesses a tag and shows current value
  ' tag1 is an I/O tag
  Set obj = Application.GetObject("IODriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  'This can also be done in a different way,
  'with no need to show Value property, which is default
  MsgBox " Tag1's current value: " & obj
End Sub
```

NOTE: Bit values (**Bit00 to Bit31** properties) are not affected by adjusting in scale, that is, they represent the bits from the value read by the equipment before conversion.

5.5.2 OPC Driver

This section contains information about **events** and **properties** of the OPC Driver object. This object does not contain methods associated to it.

5.5.2.1 Events

This section contains information about the events of the OPC Driver object.

5.5.2.1.1 OnTagRead

OnTagRead(*Tag*)

Occurs when reading a Tag, whenever a new value or error is returned by the OPC Server. That is, if Tag value or quality do not change, this event is not triggered. For this event to occur, Tag's **EnableDriverEvent** property must be enabled.

5.5.2.1.2 OnTagWrite

OnTagWrite(*Tag*, *Succeeded*, *User*)

Occurs when a writing is triggered on any OPC Driver's Tag. For this event to occur, Tag's **EnableDriverEvent** property must be enabled. If this writing is asynchronous, the **OnTagWrite** event is triggered only when the server sends a response indicating whether that writing was successful or not.

Parameters of the OnTagWrite event

NAME	DESCRIPTION
Tag	A reference to the Tag object being written. For example, users may access a Tag property using a **Tag.DocString** syntax.
Succeeded	A **Boolean** value indicating success or failure on writing.
User	Parameter that receives the user performing the writing operation. The default value of this parameter is "System". If there is no user logged in Viewer generating this event, this parameter contains the value "Anonymous". If the writing is asynchronous or if there is a failure reported in an asynchronous way, this parameter always contains the value "System".

5.5.2.2 Properties

This section contains information about the properties of the OPC Driver object.

5.5.2.2.1 CallTimeout

Specifies a time-out for an answer on any call or access to an OPC server, such as writings, creation of OPC Groups, Tag creation, removal, or browsing, changes to Tag's **Advise** mode, etc. If this time-out is exceeded, E3 considers the server as locked (unavailable) and starts the reconnection process. The value of this property cannot be negative. Configuring this property with 0 (zero) disables this time-out, causing all accesses to the OPC server to take an undefined amount of time, which

may lock the whole application if this access is synchronous. The default value of this property is 10000 (10 seconds). Applications created in previous versions, when loaded in the current version, assume the value 0 (zero) for compatibility reasons. This value must be carefully configured, so that it does not lock the whole application nor forces an unnecessary disconnection, if the OPC server is actually taking too long to respond to certain requests.

5.5.2.2.2 Compatibility

Allows controlling the usage of OPC's default interfaces by E3's OPC client. The available options are the following:

- **0 - AnyVersion**: Normal operation (recommended), the OPC Driver communicates with both DA 2.0x and 1.0a servers (preference is given to 2.0x interfaces)
- **1 - Version10A**: Forces communication with DA 1.0a standard for servers that support both DA 2.0x and 1.0a
- **2 - Version20**: Forces communication only with OPC DA 2.0x standard

This property cannot be modified when OPC client communication is enabled (both in Studio and at run time).

NOTE: Only as a last resource the Driver should be configured with a value different from **0 - AnyVersion** (default value). This property is for advanced usage only, and applies strictly to overcome a possible incompatibility situation with specific OPC servers.

5.5.2.2.3 ConnectionTimeout

Specifies a maximum time-out to establish a connection with an OPC server, including all connection steps, even steps before accessing the server itself, such as access to the **OPCENUM** service or to Windows Registry. The value of this property cannot be negative. Configuring this property with 0 (zero) disables this time-out, causing this limit to be the actual error's return time by the services needed to access the OPC server. The default value of this property is 10000 (10 seconds). Applications created in previous versions, when loaded in the current version, assume the value 0 (zero) for compatibility reasons.

5.5.2.2.4 ReconnectPeriod

Controls the period of connection to the OPC Server. If connection is lost, the OPC Driver stops and restarts until this action returns success. The period is configured in milliseconds and when its value is set to 0 (zero), reconnection is disabled. Because the OPC Driver is stopped and started, **BeforeStart** and **AfterStop** events are generated. When connection is lost, all related Tags are disconnected from their

current status (bad, quality, or null value). Example:

```
Sub OPCDriver1_AfterStart()
  Application.GetObject("OPCDriver1.OPCGroup1")._
    ReconnectPeriod = 0
End Sub
```

5.5.2.2.5 ServerId

A Determines the server to which the OPC Driver must connect. Although the default value of this property is empty, if this value is empty the OPC Driver is not going to connect. This property can only be modified when the OPC Driver is not connected. Example:

```
Sub CommandButton1_Click()
  Set Opc = Application.GetObject("OPCDriver1")
  Opc.Deactivate
  Opc.ServerId = "ElipseSCADA.OPCSvr.1"
  Opc.ServerMachine = "\\server2"
  Opc.Activate
End Sub
```

5.5.2.2.6 ServerMachine

A This property determines the address of the station where the OPC server is running. For applications running locally, this property may remain blank (default). Otherwise, users must specify the path (for example, "\\ServerName"). This property can only be modified when the OPC Driver is disconnected. Example:

```
Sub CommandButton1_Click()
  Set Opc = Application.GetObject("OPCDriver1")
  Opc.Deactivate
  Opc.ServerId = "ElipseSCADA.OPCSvr.1"
  Opc.ServerMachine = "\\server2"
  Opc.Activate
End Sub
```

5.5.2.2.7 ServerName

A This property returns the OPC server's name or description. This property is different from the **ServerID** property, which is a code. Example:

```
Sub OPCDriver1_AfterStart()
  MsgBox _
    Application.GetObject("OPCDriver1.OPCGroup1").ServerName
End Sub
```

5.5.2.2.8 ServerStatus

Determines the OPC Server's connection status. The available options are described on the next table.

Available options for ServerStatus

OPTION	DESCRIPTION
-1 - ServerStatus_Unknown	The OPC Driver is connected to the OPC server but either status is not informed or the OPC client has its **ReconnectPeriod** property set to 0 (zero).
0 - ServerStatus_NotConnected	The OPC Driver is not connected to the OPC Server. This happens when, for example, the OPC Driver is not active, or a connection has not been established yet for any reason.

The following values are only informed when the **ReconnectPeriod** property is different from 0 (zero). This time period specifying a status is retrieved from the server. In case this status is not correctly informed, this property can maintain its value in -1, or the disconnection can be detected in this case, which brings the **ServerSatus** property to 0 (zero). Values are based on the five default status types defined for OPC servers.

Available options for ReconnectPeriod different from zero

OPTION	DESCRIPTION
1 - ServerStatus_Running	The server is executing normally.
2 - ServerStatus_Failed	The server is not executing. An unspecified error occurred on the server.
3 - ServerStatus_NoConfig	The server is executing, but without information on its configuration.
4 - ServerStatus_Suspended	The server was temporarily suspended.
5 - ServerStatus_Test	The server is in test mode.

Example:

```
Sub CommandButton1_Click()
  Dim status
  status =  Application.GetObject("OPCDriver1").ServerStatus
  MsgBox "Driver status is " & status
  Select Case status
    Case -1
      MsgBox "OPC Driver is connected to the OPC Server_
        but its status was not informed."
    Case 0
      MsgBox "OPC Driver is not connected to an OPC Server."
    Case 1
      MsgBox "The OPC Server is running normally."
    Case 2
      MsgBox "The OPC Server is not running."
    Case 3
```

```
      MsgBox "The OPC Server is running without information_
        on its configuration."
    Case 4
      MsgBox "The OPC Server has been temporarily suspended."
    Case 5
      MsgBox "The OPC Server is running in Test Mode."
  End Select
End Sub
```

NOTE: To obtain a behavior equivalent to a **Boolean** property, it is recommended to use **ServerStatus** different from 0 (zero). This basically differentiates only between having or not having a connection, without considering more specific server statuses. In addition, this expression does not depend on using the **ReconnectPeriod** property different from 0 (zero).

5.5.2.2.9 WriteFeedbackMode

This property allows controlling feedback from Tag writings. The configuration options of this property are described on the next table.

Available options for the WriteFeedbackMode property

OPTION	DESCRIPTION
0 - wfWaitNextRead	After writing, waits normally for the next reading.
1 - wfImmediateReadAfterWrite	Forces an asynchronous reading from a device right after each writing.
2 - wfTrustWriteSuccess	The written value is assumed by the Tag immediately, if the writing succeeded.

The default value of this property is **0 - wfWaitNextRead** for applications created before the implementation of this property, and **1 - wfImmediateReadAfterWrite** for applications created after its implementation.

NOTES:

- OPC Driver's **WriteFeedbackMode** property cannot be changed while this object is active.
- In the **2 - wfTrustWriteSuccess** option, for asynchronous writings, the value is assumed in the Tag right after scheduling that writing, if the operation succeeded. However, if that writing fails later, the value in the Tag may be incorrect. For synchronous writings, this value is assumed right after that writing finishes, if succeeded.
- Please check I/O Driver's **WriteFeedbackMode** property, which has a similar behavior.

5.5.2.3 OPC Group

This section contains information about **methods** and **properties** of the OPC Group object. This object does not contain methods associated to it.

5.5.2.3.1 Methods

This section contains information about the methods of the OPC Group object.

5.5.2.3.1.1 Refresh

Refresh(*Source*)

Forces the server to resend values from all OPC Group Tags that have their readings enabled, whether they have changed their values or not. The *Source* parameter determines the argument of OPC Driver's data source. If the informed value is **1** (**RefreshFromCache**), sent values are server's cache values. Otherwise, if the informed value is **2** (**RefreshFromDevice**), values are updated in server's cache before sending. For this method to work, the OPC Group's **Enable** property, as well as reading at least one Tag of the OPC Group, must be enabled. For more information about this mechanism of enabling readings (the **Advise** mode), please check Tag's **AllowRead** and **AdviseType** properties.

5.5.2.3.2 Properties

This section contains information about the properties of the OPC Group object.

5.5.2.3.2.1 BlockMode

◩ This property determines the behavior when activating or deactivating an OPC Group. when this property is configured to True, communication of Tags of the OPC Group starts at once. This usually improves performance (short activation time), by minimizing the number of calls to the OPC server. When this property is configured to False, its behavior is activating communication for each Tag of the OPC Group individually (according to the normal activation sequence of objects). With it, for example, the first Tag of the OPC Group (in the order seen in the Organizer) communicates before the last Tag. Altough slower, this activation mode of an OPC Block can be an advantage when trying to perform an operation (a Tag writing, for example) in a script of a Tag's **OnStartRunning** event. The deactivation occurs in an analogous way. When the value of this property is True, the deactivation of communication of OPC Group Tags occurs at once, when finishing the deactivation of the whole OPC Group. In case this property is configured to False, deactivation of communication occurs individually for each Tag (according to the normal activation sequence of objects).

5.5.2.3.2.2 DeadBand

This property allows adjusting a minimum variation level of an OPC Tag, so that it can be updated. This property only applies to OPC Group Tags that are considered as of analog type by the OPC Server to which the OPC Driver is connected. The valid interval for this property is between 0 and 100%. A value of 0 (zero) for this property means that any variation in the value of an OPC Group Tag implies in updating this OPC Group. This percentage applies to each Tag according to their engineering limits (defined in the OPC server). To update a Tag, the following expression must be true (evaluated in the OPC server):

```
Abs(Previously_stored_value - Current_value) > (DeadBand / 100)
* Abs(Upper_limit - Lower_limit)
```

The default value of this property is 0 (zero).

5.5.2.3.2.3 Enable

This property enables updating Tags inside an OPC Group. If this property is False, no Tag inside the OPC Group is updated. Otherwise, Tags with their **AllowRead** property set to True and in **Advise** mode (for more information, please check the **AdviseType** property) are kept updated according to the update time (the **Scan** property) and OPC Group's dead band (the **DeadBand** property). If this property is False, it is not possible to use the OPC Group's **Refresh** method.

5.5.2.3.2.4 RealScan

Scan time effectively used by the OPC server.

5.5.2.3.2.5 Scan

Specifies the update's scan time of OPC Group Tags that is used by the server. This property is represented in milliseconds and it can be modified after starting communication, and it is used only when the **Enable** property is set to True.

When configuring this property in several Tags in the application, users should increase the value of this property on those OPC Group Tags that do not vary very much in the device, enabling other OPC Group Tags with higher priority to be read more frequently, thus enhancing application's performance and responsiveness. The default value of this property is 1000.

5.5.2.3.3 OPC Block

This section contains information about **events**, **methods**, and **properties** of the OPC Block object.

5.5.2.3.3.1 Events

This section contains information about the events of the OPC Block object.

OnRead

OnRead()

This event occurs when an OPC Block value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying data in the OPC Block (the **Bit00 to Bit31**, **Quality**, **RawValue**, **TimeStamp**, and **Value** properties of any OPC Block Element).

5.5.2.3.3.2 Methods

This section contains information about the methods of the OPC Block object.

Write

Write()

Writes the OPC Block's current value to the device. For more information, please check the Driver's documentation. This method returns a **Boolean** indicating whether this operation succeeded or not.

WriteEx

WriteEx([*Value*[, *SyncWrite*]])

Writes a value to the device. All its parameters are optional. If omitted, the behavior of this method is the same as the **Write** method. This method returns a **Boolean** indicating whether this operation succeeded or not. The *Value* parameter defines the value to write to the OPC Driver. Data type depends on the Driver. If omitted, assumes the OPC Block's current value. The *SyncWrite* parameter is a **Boolean** specifying whether this operation must be synchronous (True) or asynchronous (False). If omitted, uses the value specified in the OPC Block's **SyncWrite** property.

NOTE: As in the **Write** method, writings are performed independent of the value being different from the OPC Block's current value, as well as independent of the OPC Block's **AllowWrite** property being True or False. In addition, if the writing works but the OPC Block is not in scan (whether its **AllowRead** property is set to False or because it is using the **AdviseWhenLinked** option when it is not linked), the written value immediately assumes a good quality and the time stamp of the moment of writing.

5.5.2.3.3.3 Properties

This section contains information about the properties of the OPC Block object.

AdviseType

Controls the **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The Tag is updated if the OPC Block's **AllowRead** property is True and the OPC Group's **Enable** property is also True.
1 - AdviseWhenLinked	The Tag is updated only if the OPC Block's **AllowRead** property and the OPC Group's **Enable** property are True, and the Tag is associated to an active object, such as a Display on an open Screen or an enabled Alarm, among others. A Tag link for this purpose can be assigned to the following properties: **Value**, **RawValue**, **Quality**, and from **Bit00 to Bit31** of OPC Block Elements, and **Quality** and **TimeStamp** of OPC Blocks.

Example:

```
Sub CommandButton3_Click()
  MsgBox Application._
    GetObject("OPCDriver.OPCGroup.SCRIPT1").AdviseType
End Sub
```

AllowRead

Configure this property to define whether this OPC Block must or must not be read by the OPC Driver. If this property is configured to True, then the Driver automatically updates OPC Block Element's **Value** and Bits (from **Bit00 to Bit31**) properties in time spans. Otherwise, this OPC Block is not read if this property is set to False. This property can be modified at run time. Its default value is True.
Example:

```
Sub Button1_Click()
  ' Stops tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

AllowWrite

Configure this property to define whether this Tag must be automatically written when the **Value** property or any Bit property (from **Bit00 to Bit31**) is modified. If this property is True, changes are sent to the device or equipment linked to the OPC Driver, otherwise these changes are ignored. If this property is configured to True, then the Driver automatically updates **Value** and Bit (from **Bit00 to Bit31**) properties of this object, in time spans. Otherwise, this OPC Block is not read. The default value

of this property is True. Example:

```
Sub Button1_Click()
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

DataType

Read-only property. Determines the data type associated to this OPC Block, according to the next table.

Available options for the DataType property

OPTION	DESCRIPTION
0 - _Undefined	Unidimensional undefined value (Empty).
1 - _Null	Null value.
2 - _Integer	Unidimensional 16-bit signed integer value.
3 - _Long	Unidimensional 32-bit signed integer value.
4 - _Single	Unidimensional 32-bit floating point value.
5 - _Double	Unidimensional 64-bit floating point value.
6 - _Currency	Unidimensional currency value with four decimal places.
7 - _Date	Date and time value.
8 - _String	Literal value (text).
9 - _Object	Unidimensional reference value to an object.
10 - _Error	Unidimensional error code value.
11 - _Boolean	Unidimensional **Boolean** value (true or false).
12 - _Variant	Data of any type used for objects and other values to which the data type is unknown.
13 - _UnkObject	Unidimensional reference value to an object.
14 - _Decimal	Unidimensional 96-bit floating point value.
36 - _Record	Unidimensional recording value.
16 - _Char	Unidimensional 8-bit integer value.
17 - _Byte	Used to create DLLs and OLE, uses one byte in memory.
18 - _Word	Unidimensional 16-bit integer value.
19 - _Dword	Unidimensional 32-bit integer value.
20 - _LongLong	Unidimensional 64-bit signed integer value.
21 - _DDWord	Unidimensional 64-bit integer value.

OPTION	DESCRIPTION
22 - _Integer_	Integer number, ranges from -32.768 to 32.767, uses two bytes.
23 - _Uinteger	Unsigned integer value (equivalent to a **DWord**), ranges from 0 (zero) to 4294967295 (2^{32} - 1).
8194 - _ArrInteger	Unidimensional array of integer values.
8195 - _ArrLong	Unidimensional array of 32-bit signed integer values.
8196 - _ArrSingle	Unidimensional array of 32-bit floating point values.
8197 - _ArrDouble	Unidimensional array of 64-bit floating point values.
8198 - _ArrCurrency	Unidimensional array of currency values with four decimal places.
8199 - _ArrDate	Unidimensional array of date and time values.
8200 - _ArrString	Unidimensional array of literal values (text).
8201 - _ArrObject	Unidimensional array of reference values to an object.
8202 - _ArrError	Unidimensional array of error code values.
8203 - _ArrBoolean	Unidimensional array of **Boolean** values (true or false).
8204 - _ArrVariant	Array of data of any type used for objects and other values to which the data type is unknown.
8205 - _ArrUnkObject	Unidimensional array of reference values to an object.
8206 - _ArrDecimal	Unidimensional array of 96-bit floating point values.
8228 - _ArrRecord	Unidimensional array of record values.
8208 - _ArrChar	Unidimensional array of char values.
8209 - _ArrByte	Unidimensional array of bytes, which are values used to create DLLs and OLE, uses one byte in memory.
8210 - _ArrWord	Unidimensional array of 32-bit integer values.
8211 - _ArrDWord	Unidimensional array of 32-bit integer values.
8212 - _ArrLongLong	Unidimensional array of 16-bit integer values.
8213 - _ArrDDWord	Unidimensional array of 32-bit signed integer values.
8214 - _ArrInteger_	Unidimensional array of 16-bit signed integer values.

OPTION	DESCRIPTION
8215 - _ArrUInteger	Unidimensional array of unsigned integer values (equivalent to a **DWord**), ranges from 0 (zero) to 4294967295 (2^{32} - 1).

EnableDriverEvent

◩ This property is used to control the generation of the **OnTagRead** event, which occurs in the OPC Driver that contains the OPC Block. If the OPC Block's **EnableDriverEvent** property is set to True, at each reading that comes from the OPC server, whether on error or not, the **OnTagRead** event is generated on the OPC Driver that contains this OPC Block. Otherwise, this event does not occur. Also, when the **EnableDriverEvent** property is set to True, at each writing sent to the OPC server, an **OnTagWrite** event is generated in the OPC Driver that contains the OPC Block. If this writing is asynchronous, the **OnTagWrite** event is generated only when the server sends an answer indicating whether the writing succeeded or not. In this case, the event is only generated if the **EnableDriverEvent** property is set to True at that moment, and not at the moment of sending the writing. The default value of this property is False.

ItemID

A This property determines the path identifying the OPC Block on the server to which the OPC Driver connects. Definition of this path is flexible and depends on the specific server. Usually, servers specify an ID space with hierarchical items, such as **ParentItem.ChildItem.Tag1**. The **ItemID** property works as a unique key for data, considering *where* or *what* allows an OPC server to connect to a data source. Although its default value is empty, users must specify a value for this OPC Block to be valid.

Quality

9 This property informs the quality of the value contained in the **Value** property. Every time the OPC Driver attributes a new value to the OPC Block, it also configures the quality of that data. This is a read-only property. The default value of this property is 0 (zero, bad quality).

NOTE: For more information about quality, please check the topic **Quality** on **E3 User's Manual**.

Size

9 Defines the size of OPC Block's set of values. Please check Driver's documentation for information about the limit for this property, according to *B1* to *B4* parameters. By creating Elements for the OPC Block, it allows both accessing the values read and also allows writing values to the device. This property cannot be modified once communication has started. The default value of this property is 0 (zero). Example:

```
Sub OPCBlock1_OnStartRunning()
```

```
  Size = 12
End Sub
```

SyncWrite

This property determines the type of writing used by an OPC Block. If this property is set to True, this writing is synchronous, that is, the OPC Driver waits for the result of a writing from the server. Otherwise, this writing is asynchronous, that is, OPC Block's value is sent and OPC Driver processing continues immediately. The default value of this property is False.

NOTE: In asynchronous mode (property disabled), communication performance tends to be better, but in synchronous mode (property enabled), success of a writing operation is verified and informed.

TimeStamp

This property is updated whenever there is a change in value or status in either **Value** or **Quality** properties. This property informs a date and time linked to OPC Block's value, as well as to OPC Block's quality. This is a read-only property. The default value of this property is "00:00:00".

5.5.2.3.3.4 OPC Block Element

This section contains information about **events** and **properties** of the OPC Block Element object. This object does not contain methods associated to it.

Events

This section contains information about the events of the OPC Block Element object.

OnRead

OnRead()

Occurs when an OPC Block Element's value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying any data in the OPC Block Element (the **Bit00 to Bit31**, **Quality**, **RawValue**, or **Value** properties).

Properties

This section contains information about the properties of the OPC Block Element object.

Bit00 to Bit31

These properties represent all 32 bits of an OPC Block Element's **Value** property, where **Bit00** is the least significant bit and **Bit31** is the most significant bit. Modifying each one of these bits implies in changing the OPC Block Element's **Value** property and vice versa, but this only happens when the **UseBitFields** property is set to True. The default value of these properties is False.

DeviceHigh

This property defines the highest value reached by an OPC Block Element in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

DeviceLow

This property defines the lowest value reached by an OPC Block Element in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EnableScaling

This property enables or disables the scale of values for a value sent or received from a device. If this property is set to True, all values from the device receive a scale adjustment according to **DeviceHigh**, **DeviceLow**, **EUHigh**, and **EULow** properties before setting it to the **Value** property. The same occurs when a writing is

needed, when the value in the **Value** property receives a scale adjustment (but without changing the **Value** property) and later it is sent to the device. If the **EnableScaling** property is set to False, no scale adjustment is performed in neither way (writing and reading). The default value of this property is False. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EU

A Identifies the engineering unit represented by this value, such as degrees, meters, etc. Example:

```
Sub Element_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

9 Defines the highest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
```

```
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EULow

Defines the lowest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of an
  ' Element's temperature
  ' ranging from 0 to 255 on OPC server,
  ' but it actually means
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

Index

Use this property to specify the position that an OPC Block Element occupies among the Elements configured in the OPC Block's **Size** property where it is inserted. This property's accepts values from 0 (zero) to the value defined by **Size** minus one. For example, configuring an OPC Block Element's **Size** property to 20, the maximum valid number for the **Index** property is 19 and the minimum is 0 (zero). This property can be changed after starting communication. The default value of this property is 0 (zero), but when mapping Elements of an OPC Block, Studio automatically sets the **Index** property to the specified value. Example:

```
Sub Element1_OnRead()
  MsgBox Index
End Sub
```

Quality

This property represents the quality status of a value in the **Value** property.

NOTE: For more information on quality, please check the topic **Quality** on **E3 User's Manual**.

RawValue

Accesses the original value of an OPC Block Element previous to the scale, that is, regardless of **EnableScaling** property's configuration. Therefore, if this property is set

to False, the **Value** and **RawValue** properties have the same behavior.

UseBitFields

If this property is set to True, every time the value in the **Value** property changes, it updates all bits referring to the **Bit00 to Bit31** properties. Likewise, it updates the value in the **Value** property every time any of the **Bit00 to Bit31** properties change their values, and later send it to the device if the OPC Block's **AllowWrite** property is set to True. Otherwise, if this property is set to False, bits do not change nor lead to any modification. This property can be updated after starting communication. The default value of this property is False.

Value

Updated whenever a new value is read from the OPC server, according to OPC Block's **ItemID** property specifications where this OPC Block Element is inserted, and also considering the **Index** property, which specifies the position of an Element in an OPC Block array. This property's data type (integer, floating point, or text) depends on the Driver to which it is associated and its configuration.

This property is only updated this way if the **AllowRead** property of the OPC Block to which the OPC Block Element belongs is set to True, and according to a scan time defined in the **Scan** property of the OPC Group that contains the OPC Block. If the OPC Block's **AllowWrite** property is set to True, users can write values to the device simply by attributing a new value to the **Value** property.

This is also the default property of an OPC Block Element. Therefore, a reference by value to an OPC Block Element does not need necessarily to explicit its **Value** property to have access to its value. If this property is not being updated, check if the **Index** property is correctly configured (its value must be between zero and the size of the OPC Block minus one). Example:

```
Sub Button1_Click()
  ' Accesses an Element and shows its current value
  ' elm1 is an OPC Block Element object
  Set obj = Application.GetObject_
    ("OPCDriver1.Group1.OPCBlock1.elm1")
  MsgBox "Elm1's current value: " & obj.Value
  ' This can also be performed in a different way,
  ' without displaying the Value property, which is default
  MsgBox "Elm1's current value: " & obj
End Sub
```

5.5.2.3.4 OPC Tag

This section contains information about **events**, **methods**, and **properties** of the OPC Tag object.

5.5.2.3.4.1 Events

This section contains information about the events of the OPC Tag object.

OnRead

OnRead()

Occurs when an OPC Tag's value is received from an OPC Server. Use the **OnRead** event when there is a need to perform an operation right after modifying any data in the OPC Tag (the **Bit00 to Bit31**, **Quality**, **RawValue**, **TimeStamp**, or **Value** properties).

5.5.2.3.4.2 Methods

This section contains information about the methods of the OPC Tag object.

Write

Write()

Performs a writing of OPC Tag's current value to the device. For more information, please check the Driver's documentation. This method returns a **Boolean** indicating whether this operation succeeded or not.

WriteEx

WriteEx([*Value*[, *SyncWrite*]])

Writes a value to the device. All its parameters are optional. If omitted, the behavior of this method is the same as the **Write** method. This method returns a **Boolean** indicating whether this operation succeeded or not. The *Value* parameter defines the value to write to the OPC Driver. Data type depends on the Driver. If omitted, assumes the OPC Tag's current value. The *SyncWrite* parameter is a **Boolean** specifying whether this operation must be synchronous (True) or asynchronous (False). If omitted, uses the value specified in the OPC Tag's **SyncWrite** property.

NOTE: As in the **Write** method, writings are performed independent of the value being different from the OPC Tag's current value, as well as independent of the OPC Tag's **AllowWrite** property being True or False. In addition, if the writing works but the OPC Tag is not in scan (if its **AllowRead** property is set to False or because it is using the **AdviseWhenLinked** option when it is not linked), the written value immediately assumes a good quality and the time stamp of the moment of the writing.

5.5.2.3.4.3 Properties

This section contains information about the properties of the OPC Tag object.

AdviseType

Controls the **Advise** mode. The available options are described on the next table.

Available options for the AdviseType property

OPTION	DESCRIPTION
0 - AlwaysInAdvise	The OPC Tag is kept updated if its **AllowRead** property is set to True, and the OPC Group's **Enable** property is also set to True.
1 - AdviseWhenLinked	The OPC Tag is only updated if the OPC Group's **AllowRead** and **Enabled** property are set to True, and the Tag is linked to an active object, such as a Display on an open Screen or an enabled Alarm, among others. A Tag link for this purpose can be performed in the following OPC Tag properties: **Value**, **RawValue**, **TimeStamp**, **Quality** and **Bit00 to Bit31**.

Example:

```
Sub CommandButton3_Click()
  MsgBox Application._
    GetObject("OPCDriver.OPCGroup.OPCTag1").AdviseType
End Sub
```

AllowRead

Defines whether this OPC Tag should be read or not by the OPC Driver. If this property is set to True, the Driver automatically updates **Value** and Bit (from **Bit00 to Bit31**) properties of this object in time spans. Otherwise, this OPC Tag is not read. This property can be modified at run time. The default value of this property is True. Example:

```
Sub CommandButton1_Click()
  ' Stops a Tag reading
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowRead = False
End Sub
```

AllowWrite

Defines whether this OPC Tag should be written automatically or not when the **Value** or any Bit (from **Bit00 to Bit31**) properties are modified. If this property is set to True, these modifications are sent to the device associated to the OPC Driver. Otherwise, these modifications are ignored. The default value of this property is True. Example:

```
Sub Button1_Click()
  ' Disables a Tag writing
  Set obj = Application.GetObject("Driver1.tag")
  obj.AllowWrite = False
End Sub
```

Bit00 to Bit31

These properties represent all 32 bits of an OPC Tag's **Value** property, where **Bit00** is the least significant bit and **Bit31** is the most significant bit. Modify each one of these bits implies in changing the OPC Tag's **Value** property and vice-versa, but this only happens when the **UseBitFields** property is set to True. The default value of this property is False.

NOTE: Bit values (the **Bit00** to **Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

DataType

Read-only property. Determines the data type associated to this OPC Tag. Please check the table **Available options for the DataType property** on OPC Block's **DataType** property for all available values for this property.

DeviceHigh

This property defines the highest value reached by this OPC Tag in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

DeviceLow

This property defines the lowest value reached by this OPC Tag in the device. This property is used to perform a scale adjustment of the value from a device before setting it to the **Value** property. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this

property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

EnableDriverEvent

◩ This property is used to control the generation of the **OnTagRead** event, which occurs in the OPC Driver that contains the OPC Tag. If the Tag's **EnableDriverEvent** property is set to True, at each reading that comes from the OPC Server, whether on error or not, the **OnTagRead** event is generated on the OPC Driver that contains this Tag. Otherwise, this event does not occur. Also, when the **EnableDriverEvent** property is set to True, at each writing sent to the OPC Server, an **OnTagWrite** event is generated in the OPC Driver that contains the OPC Tag. If this writing is asynchronous, the **OnTagWrite** event is generated only when the server sends an answer indicating whether the writing succeeded or not. In this case, the event is only generated if the **EnableDriverEvent** property is set to True at that moment, and not at the moment of sending the writing. The default value of this property is False.

EnableScaling

◩ If this property is set to True, all values from the device receive a scale adjustment, according to **EUHigh** and **EULow** properties before setting it to the **Value** property. If this property is set to False, no scale adjustment is performed in neither way (writing and reading). The default value of this property is False. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
```

```
End Sub
```

EU

A Identifies the engineering unit represented by the value, such as degrees, meters, etc. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

EUHigh

9 Defines the highest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 1000. Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

EULow

9 Defines the lowest value to set to the **Value** property, adjusting the scale to the device's value before setting it. Likewise, at the moment of a writing the inverse operation is performed before sending the value to the Driver. This conversion only occurs when the **EnableScaling** property is set to True. The default value of this property is 0 (zero). Example:

```
Sub Tag_OnStartRunning()
  ' Performs a scale adjustment of a
  ' Tag's temperature
  ' ranging from 0 to 255 on PLC,
  ' but it actually means
  ' 0 to 100 Celsius degrees
  EU = "Celsius degrees"
  EnableScaling = True
  DeviceHigh = 255
  DeviceLow = 0
  EUHigh = 100
  EULow = 0
End Sub
```

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

ItemID

A This property determines the path that identifies the OPC Tag on the OPC server to which the OPC Driver connects. Definition of this path is flexible and depends on the specific server. Usually, servers specify an ID space with hierarchical items, such as **ParentItem.ChildItem.Tag1**. The **ItemID** property works as a unique key for data, considering *where* or *what* allows an OPC server to connect to a data source. Although its default value is an empty **String**, users must specify a value for this OPC Tag to be valid.

Quality

9 This property informs the quality of the value contained in the **Value** property. Each time the OPC Driver attributes a new value to the OPC Tag, it also configures the quality of that data. This is a read-only property. The default value of this property is 0 (zero, bad quality).

NOTE: For more information about quality, please check the topic **Quality** on **E3 User's Manual**.

RawValue

v Accesses the original value of the OPC Tag previous to the scale, that is, regardless of **EnableScaling** property's configuration. Therefore, if this property is set to False, the **Value** and **RawValue** properties have the same behavior.

SyncWrite

◩ This property determines the type of writing used by an OPC Tag. If this property is set to True, this writing is synchronous, that is, the OPC Driver waits for the result of a writing from the server. Otherwise, this writing is asynchronous, that is, OPC Tag's value is sent and OPC Driver processing continues immediately. The default value of this property is False.

NOTE: In asynchronous mode (property set to False), communication performance tends to be better, but in synchronous mode (property set to True), success of a writing operation is verified and informed.

TimeStamp

UThis property is updated whenever there is a change in value or status in either **Value** or **Quality** properties. This property informs a date and time linked to OPC Tag's value, as well as to OPC Tag's quality. This is a read-only property. The default value of this property is "00:00:00".

UseBitFields

If this property is set to True, every time the value in the **Value** property changes, it updates all bits referring to the **Bit00 to Bit31** properties. Likewise, it updates the value in the **Value** property every time any of the **Bit00 to Bit31** properties change their values, and later send it to the device if the OPC Tag's **AllowWrite** property is set to True. Otherwise, if this property is set to False, bits do not change nor lead to any modification. This property can be updated after starting communication. The default value of this property is False.

NOTE: Bit values (the **Bit00 to Bit31** properties) are not affected by scale adjustments, that is, they represent the bits of the value read from a device before conversion.

Value

This property is updated whenever performing a new valid reading of a value from a device using its configuration, but the data type of this variable (integer, floating point, or text) depends on the OPC Driver to which it is linked and on its configuration. This property is only updated this way if the **AllowRead** property is set to True and there is no communication errors (in this case only the **Quality** and **TimeStamp** properties are updated), but according to a scan time defined on the OPC Group to which the OPC Tag belongs. Another way of using this property is writing values to the device. Just set a new value to the **Value** property or any of the **Bit00 to Bit31** bits, as long as the **AllowWrite** property is set to True. This property is also the standard property of an OPC Tag object. Thus, a reference by value to an OPC Tag object does not need to explicit use the **Value** property to access its value. The default value of this property is empty. Example:

```
Sub Button1_Click()
  ' Accesses a Tag and shows its current value
  ' tag1 is an OPCTag-type object
  Set obj = Application._
    GetObject("CommDriver1.tag1")
  MsgBox "Tag1's current value: " & obj.Value
  ' Without showing the Value property,
   which is the default one
  MsgBox "Tag1's current value: " & obj
End Sub
```

5.5.3 OPC UA Driver

This section contains information about **properties** of the OPC UA Driver object. This object does not contain events nor methods associated to it.

5.5.3.1 Properties

This section contains information about the properties of the OPC UA Driver object.

5.5.3.1.1 EndPointURL

A Read and write property that specifies the path (endpoint) of the OPC UA server to which the client connects. This property cannot be changed while the connection is active.

5.5.3.1.2 Password

A Read an write property that specifies the user's password used in the connection with an OPC UA server. This property is used together with the **UserName** property and its default value is an empty **String**.

NOTE: This property can be changed while the communication is active, but that change is only effective when the OPC UA Client is restarted.

5.5.3.1.3 SecurityMode

Read and write property that specifies the security mode used when connecting to an OPC UA server. This property cannot be changed while the connection is active. Possible values for this property are the following:

- **1 - usmNone**: Does not use any security mode in the connection
- **2 - usmSign**: Uses authentication in the connection
- **3 - usmSignAndEncrypt**: Uses authentication and encryption in the connection

This property is used together with the **SecurityPolicy** property to determine the type of security of the connection. The default value of this property is 1 (**usmNone**), that is, no security.

NOTE: If the **SecurityMode** property is different from **usmNone**, the OPC UA server must accept the certificate from E3 client's instance. The way a certificate is accepted or recognized by an OPC UA server depends on the server.

5.5.3.1.4 SecurityPolicy

Read and write property that specifies the security policy (encription) used when connecting to an OPC UA server. This property cannot be changed while the communication is active. Possible values for this property are the following:

- **1 - uspNone**: Does not use encryption in the connection
- **2 - uspBasic128Rsa15**: Uses the RSA algorithm with a 128-bit key in the connection
- **3 - uspBasic256**: Uses the AES algorithm with a 256-bit key in the connection

This property is used together with the **SecurityMode** property to determine the type of security of the connection. The default value of this property is 1 (**uspNone**), that is, no security policy.

5.5.3.1.5 TimeoutCall

9 Read and write property that specifies a time limit for a call to an OPC UA server, in milliseconds. This property cannot be changed while the communication is active, its value must be greater than 0 (zero), and its default value is 10000 (10 seconds).

NOTE: In the OPC UA standard, several calls allow to continue the communication, with multiple responses from the server. So, this time ends up being the limit time for a response from the OPC UA server.

5.5.3.1.6 TimeoutConnection

9 Read and write property that specifies the time limit of a connection, in milliseconds. The default value of this property is 10000 (10 seconds) and its value must be greater than 0 (zero). This property cannot be changed while the communication is active.

5.5.3.1.7 TimeoutSession

9 Read and write property that specifies a time limit for renewing a communication session from an E3 client to an OPC UA server, in milliseconds. The default value of this property is 600000 (10 minutes) and this value must be greater than 0 (zero). This property cannot be changed while the communication is active.

5.5.3.1.8 UserName

A Read and write property that specifies the user's name used in the OPC UA server's connection. This property is used together with the **Password** property. The

default value of this property is an empty **String**.

NOTE: This property can be changed while the communication is active, but that change is only effective when the OPC UA Client is restarted.

5.6 Data Server

This section contains information about events, methods, and properties of the following objects: **Query**, **Alarm Filter**, **Data Folder**, **Counter Tag**, **Demo Tag**, **Internal Tag**, and **Timer Tag**.

5.6.1 Query

This section contains information about **events**, **methods**, and **properties** of the Query object.

5.6.1.1 Events

This section contains information about the events of the Query object.

5.6.1.1.1 OnAsyncQueryFinish

OnAsyncQueryFinish(*Recordset*, *Error*)

Occurs when the **GetAsyncADORecordset** method returns. The *Recordset* parameter is the ADO Recordset object generated by the Query, and the *Error* parameter is a **Boolean** that, when True, indicates that the object could not be generated. Example:

```
Sub query1_OnAsyncQueryFinish(Recordset, Error)
  MsgBox "Returned " + CStr(Recordset.RecordCount) + " records"
End Sub
```

5.6.1.2 Methods

This section contains information about the methods of the Query object.

5.6.1.2.1 AddField

AddField(*Name*[, *Table*])

The **AddField** method adds a new field from the table in the Query. The *Name* parameter determines the name of the new field to add to the Query. The *Table* parameter represents the name of the table to which the field belongs. This method was developed only for compatibility with E3Chart's old Query object. Example:

```
Sub Button1_Click()
  Screen.Item("E3Browser").Item("Query").AddField "Field1"
End Sub
```

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.2 AddStorageTag

AddStorageTag(*Name*, *FieldType*)

Adds a Tag from the Storage to the Query. The *Name* parameter receives the name of the Tag to add. The *FieldType* parameter indicates the Tag type (**0: Double**, **1: Bit**, **2: String**, **3: Integer**). Returns a **Boolean** indicating whether the operation was successful or not.

NOTE: This method will fail if the **QueryType** property is configured to **0 - qtDBServer**.

5.6.1.2.3 AddTable

AddTable(*Name*)

Adds a table from the database to the Query. The *Name* parameter determines the name of the table to add.

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.4 AddUaField

AddUaField(*Name*[, *Alias*[, *Function*]])

This method adds, at run time, a field to a query configured as an **OPC UA**-type. The methods returns True if successful and False on failure. The parameters of this method are the following:

- **Name**: A **String** with the path (Column) of the field
- **Alias**: A **String** with the title of the field. If this parameter is omitted, assumes an empty **String**
- **Function**: A **String** with the type of aggregation function of the field, in case a **Processed Data**-type query is used. If this parameter is omitted, assumes the value **Interpolative**

In a **Raw Data**-type query (the **UaQueryType** property equal to zero), this method fails if there is already a field defined. This method is only effective if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**). Otherwise, it returns False.

5.6.1.2.5 Execute

Execute(*ImmediateExecute*)

The **Execute** method executes a SQL command that does not have a return (such as **DELETE**, **UPDATE**, or **INSERT**), configured in the Query's **SQL** property. The *ImmediateExecute* parameter indicates whether the operation passes through database operation queues (.e3i and .e3o files) before it reaches the database (if False), or it is sent directly to the database (if True). The advantage of using a Query to execute commands is using variables, such as in a simple query.

Example of SQL commands:

```
DELETE FROM test WHERE cod > 10
UPDATE test SET cod = 10 WHERE cod > 10
INSERT INTO test (cod) VALUES(10)
```

Example:

```
Sub CommandButton1_Click()
  Screen.Item("Query1").Execute
End Sub
```

5.6.1.2.6 GetADORecordSet

GetADORecordSet()

The **GetADORecordSet** method returns an **ADO**-type (*ActiveX Data Object*) Recordset as the result of executing the configured Query. Example:

```
Sub Button1_Click()
  Set rec = Screen.Item("Query1").GetADORecordset()
  strDates = " "
  i = 0

  ' Displays a message with the first 10 records
  ' of the E3TimeStamp column
  While (NOT rec.EOF AND i < 10)
    strDates = strDates & CStr(rec("E3TimeStamp")) & _
      Chr(10) & Chr(13)
    i = i + 1
    rec.MoveNext()
  Wend
  MsgBox strDates
End Sub
```

NOTE: For more information about the ADORecordset object returned by this method, please check Microsoft documentation at *http://msdn.microsoft.com/en-us/library/ms675841(VS.85).aspx*.

5.6.1.2.7 GetAsyncADORecordSet

GetAsyncADORecordSet()

Creates a Query and, when it finishes, generates the object's **OnAsyncQueryFinish** event, passing the recordset generated by the Query to this event.

5.6.1.2.8 GetE3QueryFields

GetE3QueryFields()

The **GetE3QueryFields** method returns a Fields Collection (columns) of a Query. Every item on that Collection has properties that can be modified, as described on topic **Query Field**. Example:

```
Sub Button1_Click()
  ' Traverses the Fields Collection,
  ' displaying the fields on a message box,
  ' and also setting them as visible
  ' on Query's configuration

  Set Browser = Screen.Item("E3Browser")
  Set Query = Browser.Item("Query")
  Set Fields = Query.GetE3QueryFields()
  For Each field In Fields
    MsgBox CStr(field.TableName) & "-" & CStr(field.ColumnName)
    field.Visible = True
  Next
  ' Performs E3Browser's Query again, which has been
  ' just modified, so that all fields appear.
  Browser.RetrieveE3QueryFields()
  Browser.Requery()
End Sub
```

NOTE: To use this method, the Query must be previously created at design time.

5.6.1.2.9 RemoveField

RemoveField(*FieldName*[, *Table*])

The **RemoveField** method removes a field previously included in a Query. The *FieldName* parameter determines the name of the field to be removed from the Query. The *Table* parameter represents the name of the table to which the field belongs. This method was developed just to keep compatibility with E3Chart's old Query object. Example:

```
Sub CommandButton1_Click()
  Screen.Item("E3Browser").Item("Query").RemoveField "Field1"
End Sub
```

5.6.1.2.10 RemoveStorageTag

RemoveStorageTag(*Name*)

Removes a previously-configured Tag in the Query. The *Name* parameter indicates the Tag's name. Returns a **Boolean** indicating whether the operation was successful or not.

NOTE: This method will fail if the **QueryType** property is configured to **0 - qtDBServer**.

5.6.1.2.11 RemoveTable

RemoveTable(*TableName*)

Removes a table from the Query. The *TableName* parameter determines the name of the table to remove.

NOTE: This method will fail if the **QueryType** property is configured to **1 - qtStorage**.

5.6.1.2.12 RemoveUaField

RemoveUaField(*Name*)

This method removes the field with the specified name, if it exists. The *Name* parameter is a **String** with the name (Column) of the field. The method returns True if it is successful and False on failure. This method is only effective if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**). Otherwise, it returns False.

5.6.1.2.13 SetVariableValue

SetVariableValue(*VarName*, *Value*)

The **SetVariableValue** method adjusts the value of a variable set in the Query, so that this value can be informed as a filter or parameter before being performed. The variable name (*VarName*) and the value (*Value*), which can be a number, a text or a date and time must be defined. Example:

```
Sub CommandButton_Click()
  Set cons = Screen.Item("E3Browser1").Item("Query1")
  InitialDate = now - 1
  FinalDate = now
  cons.SetVariableValue "IniDate", InitialDate
  cons.SetVariableValue "EndDate", FinalDate
End Sub
```

5.6.1.3 Properties

This section contains information about the properties of the Query object.

NOTE: It is not advisable accessing these properties directly via scripts. Users should access the Query object, passing parameters through the **SetVariableValue** method, and then modifying filters or fields through the collection returned by the **GetE3QueryFields** method.

5.6.1.3.1 CursorLocation

Defines where the Query is generated and handled, from a DBMS (Database Management System) point of view. The available options are:

- **0 - clServer**: the Query is generated on the DBMS (server).
- **1 - clClient**: the Query is generated on the Server (client).

The default value of this property is **1 - clClient**. See also the **CursorType** property.

NOTE: This property has no effect for Databases with the **SourceType** property equal to **0 - stAccess**, because in this case the Server always generates and handles the Query. However, for **SourceType** equal to **1 - stOracle** or **2 - stSqlServer**, it is advisable to use the option **1 - clClient**. For more information, check the documentation of the **CursorLocation** property and the **CursorLocationEnum** enumeration of the ADO (*ActiveX Data Object*) object.

5.6.1.3.2 CursorType

Defines the type of a Query, according to the way data is viewed. The available options are:

Available options for CursorType

OPTION	DESCRIPTION
0 - ctKeyset	Any change in records initially returned by the Query will be visible (Default).
1 - ctStatic	No change in records initially returned by the Query will be visible.
2 - ctDynamic	All new records added to the Query are visible, in addition to the changes in records initially returned by the Query.

5.6.1.3.3 DataSource

A Indicates the Database, Storage, or OPC UA Driver object that is used in the Query. This is a read-only property, but it can be modified at run time.

5.6.1.3.4 Fields

A Text with the fields to be viewed in the Query, separated by commas. It corresponds to the argument of the **SELECT** clause in Query's SQL code. When blank, it determines that the Query must display all table fields. This is a read-only property, but it can be modified at run time.

5.6.1.3.5 FunctionSubType

Specifies the function subtype indicated by the **FunctionType** property. Only the options **1 - ftArchivedValue, 2 - ftTagAttribute**, and **6 - ftCalculatedData** have subtypes. For the other functions, **FunctionSubType** assumes **-1 - fsNoSubType**. The following table shows the possible property values, according to the chosen function in the **FunctionType** property:

Subtypes for ArchivedValue function (FunctionType = 1)

SUBTYPE	DESCRIPTION
0 - fsPreviousArchivedValue	Value stored right before the informed time stamp.
1 - fsInterpolatedArchivedValue	Calculated value, based on previous and next values.
2 - fsNextArchivedValue	Value stored right after the informed time stamp.
3 - fsExactArchivedValue	If a value is found at the exact informed time stamp.

Subtypes for TagAttribute function (FunctionType = 2)

SUBTYPE	DESCRIPTION
0 - fsTagAttributeDescription	Tag's meaning or description.
1 - fsTagAttributeSource	Path of the Tag being stored.
2 - fsTagAttributeType	Data type: **Double**, **Boolean**, or **String**.
3 - fsTagAttributeEU	Engineering unit.
4 - fsTagAttributeLowEng	Lower limit.
5 - fsTagAttributeHighEng	Upper limit.
6 - fsTagAttributeDeadBand	Dead band for recording.
7 - fsTagAttributeDeadBandUnit	Dead band unit (an absolute value or a percentage).
8 - fsTagAttributeMinRecTime	Minimum recording time (variations smaller than this interval are not recorded).
9 - fsTagAttributeMaxRecTime	Maximum recording time (the absence of variation on these intervals forces a recording).

Subtypes for CalculatedData function (FunctionType = 6)

SUBTYPE	DESCRIPTION
0 - fsTotalCalculatedData	Total of values.
1 - fsMinimumCalculatedData	Minimum value.
2 - fsMaximumCalculatedData	Maximum value.
3 - fsStandardCalculatedData	Standard deviation.
4 - fsRangeCalculatedData	Range of values.
5 - fsMeanCalculatedData	Mean of values.
6 - fsMedianCalculatedData	Median of values.

5.6.1.3.6 FunctionType

This property is valid when a Storage object is the Query's source (this is indicated by the **DataSource** property). It specifies the function that will define data generated by the Query. Some functions have subfunctions, which can be indicated in the **FunctionSubType** property. The **FunctionType** property can assume the following values:

Available options for FunctionType

OPTION	DESCRIPTION
-1 - ftNoFunction	There is no defined function. This value is not available when the **QueryType** property is configured to **1 - qtStorage**.
0 - ftLastValue	Returns the last value stored in the Database.
1 - ftArchivedValue	Returns a stored value relative to a predefined time stamp defined in the `TimeStamp` variable. The relationship is specified in the **FunctionSubType** property.
2 - ftTagAttribute	Returns a Tag attribute, defined in the **FunctionSubType** property.
3 - ftCompressedDataNValues	Returns, for a single Tag, *N* values defined in the variable `NumVals`, stored from an initial time, defined in the variable `StartTime`.
4 - ftCompressedDataStartEndTime	Returns, for a single Tag, the values stored at the interval between the variables `StartTime` and `EndTime`.
5 - ftSampledData	Returns, for one or more Tags, the interpolated values (that is, estimated) at the interval between the variables `StartTime` and `EndTime`, in fixed intervals defined by the variable `TimeInterval`.

OPTION	DESCRIPTION
6 - ftCalculatedData	Returns, for one or more Tags, the result of the math operations applied to data at the interval between variables `StartTime` and `EndTime`, in fixed intervals defined by the variable `TimeInterval`.

NOTE: Variables can also be defined at run time using the Query's **SetVariableValue** method.

5.6.1.3.7 GroupBy

A Text corresponding to the argument of the **GROUP BY** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.8 Having

A Text corresponding to the argument of the **HAVING** clause in Query's SQL code. This property is commonly used with the **GroupBy** property. This is a read-only property, but it can be modified at run time.

5.6.1.3.9 IgnoreQuality

◩ Allows indicating whether bad quality data is included or not on Query's results. This property is only effective if the Query object is using a Storage as its data source. This property can be changed by script at run time.

5.6.1.3.10 OrderBy

A Text corresponding to the argument of the **ORDER BY** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.11 QueryType

Indicates the type of query to perform. Possible values for this property are the following:

- **-1 - qtUndefined**: The Query object initially tries to execute the query on a Database object. If it fails, then it tries to execute the query on a Storage object. This is the default value for Queries created in Studio or at run time
- **0 - qtDBServer**: The Query object tries to execute the query only on a Database object
- **1 - qtStorage**: The Query object tries to execute the query only on a Storage object

- **2 - qtOpcUa**: The Query object tries to execute the query only on an OPC UA Driver object

NOTES:
- In case of manipulating Queries at run time, it is recommended to configure this property with the desired type before performing other settings on the object, specially when reusing a Query object for different data sources.
- For applications created on previous versions, when opening the application in Studio this property is automatically configured to **-1 - qtUndefined**.
- The **AddStorageTag** and **RemoveStorageTag** methods will fail if this property is configured to **0 - qtDBServer**.
- The **FunctionType** property does not accept the value **-1 - ftNoFunction** if this property is configured to **1 - qtStorage**.
- The **AddField**, **AddTable**, and **RemoveTable** methods will fail if this property is configured to **1 - qtStorage**.
- If this property is configured to **-1 - qtUndefined**, an attempt to execute the Query on a Storage object (after failing on a Database object) will fail if the **FunctionType** property is configured to **-1 - ftNoFunction** or if the **FunctionSubType** property is configured to an invalid value.

5.6.1.3.12 SQL

A Contains the SQL code specified for the Query. This is a read-only property, but it can be modified at run time.

5.6.1.3.13 Table

A Contains the tables to be queried (for example, Alarms is the alarms or events table). It corresponds to the argument of the **FROM** clause in Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.3.14 UaNamespaceArray

This property returns an object that is a **Collection of OPC UA Namespaces**, used by the fields configured in the Query. This is a read-only property.

5.6.1.3.15 UaQueryType

This is a read and write property that allows determining whether the Query is a **Raw Data** (**0: uqtRaw**) or **Processed Data** (**1: uqtProcessed**, default value) type. In Studio this property is read-only. At run time it allows configuring the type of OPC UA query, but it only accepts writings if the **QueryType** property is configured as **OPC UA** (value equal to **2: qtOpcUa**).

5.6.1.3.16 Where

A Determines the Query condition that filters the table records to be viewed, that is, only the records that meet this conditions are viewed. It corresponds to the **WHERE** argument in the Query's SQL code. This is a read-only property, but it can be modified at run time.

5.6.1.4 Fields Collection

This section contains information about **methods** and **properties** of the Query's Fields Collection object. This object does not have specific events.

5.6.1.4.1 Methods

This section contains information about the methods of the Query's Fields Collection object.

5.6.1.4.1.1 Add

Add(*NewItem*)

Adds a new Query Field object on a Query's Fields Collection, indicated by the *NewItem* parameter.

5.6.1.4.1.2 Item

Item(*Index*)

Returns a Query Field object from the Fields Collection. The *Index* parameter can be the numerical index of this Field, or its name.

5.6.1.4.1.3 RefreshUaNodeIds

RefreshUaNodeIds()

Updates all Query Fields, retrieving the OPC UA Node Identifiers (NodeIds) on the server and updating the **UaNodeId** property of the Query Fields. For this method to work, the following conditions must be met:

- The Query's **QueryType** property must be configured as **2: qtOpcUa**
- The Query must point to a valid OPC UA Driver
- The OPC UA Driver configured in the Query must be active and connected

5.6.1.4.1.4 Remove

Remove(*Index*)

Removes a Query Field object from the Query's Fields Collection. The *Index* parameter can be a **String** with the name of the Query Field, or the index of that object on the Fields Collection.

5.6.1.4.2 Properties

This section contains information about the properties of the Fields Collection object of a Query.

5.6.1.4.2.1 Count

9 Returns the amount of Query Field objects in this Collection. This is a read-only property.

5.6.1.4.3 Query Field

This section contains information about **properties** of the Query Field object. This object does not have specific events nor methods.

5.6.1.4.3.1 Properties

This section contains information about the properties of the Query Field object.

Alias

A Alias of this Field in the Query.

ColumnName

A Column's name. This name must exist on the tables added to the Query.

Criteria

A Filter that will be applied to this Field.

Function

A Function to which this Field can be passed as its parameter.

GroupBy

◩ When in True, indicates that this Field is part of a group.

OrderBy

A Sort order of Field's data. Valid values are "ASC" (ascending), "DESC" (descending), or an empty **String** (no sort order). Any other value means that this Field has no sort

order.

OrderNumber

9 Order number of this Field relative to the other Fields that compose the Query's sort order. This value will only be accepted as a number greater than 0 (zero) if the Field has a sort type (defined by the **OrderBy** property). This value must be less or equal to the number of Fields that compose the Query's sort order.

TableName

A Name of the Field's table. This table must have been added during Query's configuration.

UaNodeId

Returns an **OPC UA Node Identifier**-type object, with the configuration of the Node that identifies a field from an OPC UA query on a certain OPC UA server. In Studio this property is only represented by an icon, on the **Id** column of the Query's field configuration.

Visible

When in True, indicates that this Field is visible.

5.6.1.5 OPC UA Namespaces Collection

This section contains information about **methods** and **properties** of the OPC UA Namespaces Collection object (UaNamespaceArray). This object does not have events associated to it.

5.6.1.5.1 Methods

This section contains information about the methods of the OPC UA Namespaces Collection object.

5.6.1.5.1.1 Add

Add(*Item*)

Adds a Namespace identifier, always at the end of the Collection. The *Item* parameter is a **String** that defines the Namespace. This parameter cannot be empty.

5.6.1.5.1.2 Item

Item(*Index*)

Returns the Namespace identifier from the specified index. The *Index* parameter must be a **LONG** ranging between 0 (zero) and **Count** minus 1 (one).

5.6.1.5.1.3 Remove

Remove(*Index*)

Removes the Namespace from the specified index. Notice that this implies in changing the indexes of Namespaces greater than the removed one. It is not possible to remove indexes 0 (zero) and 1 (one). The *Index* parameter is a value (**LONG**) that identifies the Namespace to remove from the Collection, ranging between 2 (two) and **Count** minus 1 (one).

5.6.1.5.2 Properties

This section contains information about the properties of the OPC UA Namespaces Collection object.

5.6.1.5.2.1 Count

9 Returns the total amount of Namespaces in the collection. The minimum value of this property is always 2 (two), as the indexes 0 and 1 are always available.

5.6.1.5.3 OPC UA Node Identifier

This section contains information about **properties** of the OPC UA Node Identifier object (E3UaNodeId). This object does not have events nor methods associated to it.

5.6.1.5.3.1 Properties

This section contains information about the properties of the OPC UA Node Identifier object.

GUID

A Read and write property that identifies this object, in case the **Type** property is equal to **2: nitGUID**. If the **Type** property is different from 2 (two), reading this property returns an error. Writing to this property always forces the **Type** property to change to 2 (two). The value of this property is a **GUID**-type **String** (*Globally Unique Identifier*), a 128-bit value. In case this property is configured with the value "{00000000-0000-0000-0000-000000000000}", then this object is identified as `Null`.

NamespaceIndex

9 Read and write property that identifies to which Namespace the Identifier refers. This index must be between 0 (zero) and the number of Namespaces minus 1 (one) in the server. When defining a field in a Query, the index refers to the **OPC UA Namespaces Collection** defined in the Query object (which at some point may be different from the server's Namespaces Collection). Its value cannot be greater than 65535, because the OPC UA standard defines this is a 16-bit value.

Numeric

9 Read and write property that identifies this object, in case the **Type** property is equal to **0: nitNumeric**. If the **Type** property is different from 0 (zero), reading this property returns an error. Writing to this property always forces the **Type** property to change to 0 (zero). Setting this property to 0 (zero) identifies this object as `Null`.

Opaque

A Read and write property that identifies this object, in case the **Type** property is equal to **3: nitOpaque**. The value of this property is an array of bytes, that is, a **String** of characters not necessarily valid or printable. If the **Type** property is different from 3 (three), reading this property returns an error. Writing to this property always forces the **Type** property to change to 3 (three). An empty **String** in this property identifies this object as `Null`.

String

A Read and write property that identifies this object, in case the **Type** property is equal to **1: nitString**. If the **Type** property is different from 1 (one), reading this property returns an error. Writing to this property always forces the **Type** property to change to 1 (one). An empty **String** in this property identifies this object as `Null`.

Type

Read and write property that determines the type of identifier this object uses. Possible values for this property are the following:

- **0 - nitNumeric**: The identifier is a number (**LONG**)
- **1 - nitString**: The identifier is a **String**
- **2 - nitGUID**: The identifier is a **GUID**-type (*Globally Unique Identifier*) **String**
- **3 - nitOpaque**: The identifier is an array of bytes

Whenever this property changes, the value of the identifier is forced to `Null`.

5.6.2 Alarm Filter

This section contains information about **methods** and **properties** of the Alarm Filter object. This object does not have events associated to it.

5.6.2.1 Methods

This section contains information about the methods of the Alarm Filter object.

5.6.2.1.1 GetEventByIndex

GetEventByIndex(*Index*)

Returns an Event object from an **Events Collection**, specified by the *Index* parameter, which corresponds to the index of an object on that Collection. For more information about the properties of the object returned by this method, please check topic **Event - Properties**. Example of a script for this method:

```
Sub CommandButton_Click()
  For i = 0 To Screen.Item("AlarmFilter1").AlarmCount - 1
    str = ""
    set evt = Screen.Item("AlarmFilter1").GetEventByIndex(i)
    str = str & "AlarmSourceName     = " &_
      evt.AlarmSourceName & Chr(13)
    str = str & "FullAlarmSourceName = " &_
      evt.FullAlarmSourceName & Chr(13)
    str = str & "EventTime           = " &_
      evt.EventTime & Chr(13)
    str = str & "EventTimeUTC        = " &_
      evt.EventTimeUTC & Chr(13)
    str = str & "InTime              = " &_
      evt.InTime & Chr(13)
    str = str & "OutTime             = " &_
      evt.OutTime & Chr(13)
    str = str & "AckTime             = " &_
      evt.AckTime & Chr(13)
    str = str & "CurrentValue        = " &_
      evt.CurrentValue & Chr(13)
    str = str & "ActorID             = " &_
      evt.ActorID & Chr(13)
    str = str & "Area                = " &_
      evt.Area & Chr(13)
    str = str & "ConditionName       = " &_
      evt.ConditionName & Chr(13)
    str = str & "EventCategory       = " &_
      evt.EventCategory & Chr(13)
    str = str & "EventType           = " &_
      evt.EventType & Chr(13)
    str = str & "Message             = " &_
      evt.Message & Chr(13)
    str = str & "Quality             = " &_
      evt.Quality & Chr(13)
    str = str & "Source              = " &_
      evt.Source & Chr(13)
    str = str & "SubConditionName    = " &_
      evt.SubConditionName & Chr(13)
    str = str & "FormattedValue      = " &_
      evt.FormattedValue & Chr(13)
    str = str & "UserField(1)        = " &_
      evt.UserField(1) & Chr(13)
```

```
    str = str & "UserField(2)        = " &_
      evt.UserField(2) & Chr(13)
    str = str & "UserField(3)        = " &_
      evt.UserField(3) & Chr(13)
    str = str & "UserField(4)        = " &_
      evt.UserField(4) & Chr(13)
    str = str & "Severity            = " &_
      evt.Severity & Chr(13)
    str = str & "Acked               = " &_
      evt.Acked & Chr(13)
    str = str & "AckRequired         = " &_
      evt.AckRequired & Chr(13)
    str = str & "ConditionActive     = " &_
      evt.ConditionActive & Chr(13)
    str = str & "Enabled             = " &_
      evt.Enabled & Chr(13)
    str = str & "EventCLSID          = " &_
      evt.EventCLSID & Chr(13)
    MsgBox str
  Next
End Sub
```

5.6.2.2 Properties

This section contains information about the properties of the Alarm Filter object.

5.6.2.2.1 ActiveAlarms

Determines the number of active alarms on an Alarm Filter. This is a read-only property.

5.6.2.2.2 ActiveHighAlarms

Indicates the number of active alarms with **High** severity. This is a read-only property.

5.6.2.2.3 ActiveHighNACKAlarms

Indicates the number of unacknowledged alarms with **High** severity. This is a read-only property.

5.6.2.2.4 ActiveLowAlarms

Indicates the number of active alarms with **Low** severity. This is a read-only property.

5.6.2.2.5 ActiveLowNACKAlarms

9 Indicates the number of unacknowledged alarms with **Low** severity. This is a read-only property.

5.6.2.2.6 ActiveMedAlarms

9 Indicates the number of active alarms with **Medium** severity. This is a read-only property.

5.6.2.2.7 ActiveMedNACKAlarms

9 Indicates the number of unacknowledged alarms with **Medium** severity. This is a read-only property.

5.6.2.2.8 ActiveNACKAlarms

9 Indicates the number of unacknowledged alarms on an Alarm Filter. This is a read-only property.

5.6.2.2.9 AlarmCount

9 Determines the number of alarms on an Alarm Filter. This is a read-only property.

5.6.2.2.10 AlarmServer

A Name of a unique Alarm Server on an application.

5.6.2.2.11 AreaFilter

A Controls visible alarm areas on an Alarm Filter. If its value is not an empty **String**, it displays events whose Area names start with the indicated text. For example, if **AreaFilter** is equal to "Ana", it displays Area alarms such as "Analog.Production" or "Analysis", but not "Digital.Analysis" or "Digital.Production". When the **SimpleAreaFilter** property is set to True, an Alarm Area also allows using wildcard characters for filtering (* or ?) and allows multiple Area filters, separated by colons. Allowed wildcard characters are:

- **"*"**: Accepts none or any amount of characters
- **"?"**: Accepts any character
- **"#"**: Accepts any digit
- **"[ ]"**: Allows specifying a set of characters
  - **"[ab]"**: Accepts a character if it is "a" or "b"

- **"[f-h]"**: Accepts a character if it is between "f" and "h"
- **"[!cz]"**: Accepts a character if it is neither "c" nor "z"
- **"[!m-p]"**: Accepts a character if it is not between "m" and "p"

Default value of this property is an empty String, that is, no filtering by area (please check also the **CustomFilter**, **SimpleAreaFilter**, **ShowHighPriority**, **ShowMediumPriority**, and **ShowLowPriority** properties).

5.6.2.2.12 Connections

Returns a collection of Connections on an Alarm Filter. For more information about the collection returned by this property, please check topic **Collection of Connections**.

5.6.2.2.13 CustomFilter

A Allows informing a customized filter for alarms, as an expression. The following fields are available for use on this filter expression:

- **Acked (Boolean)**: Indicates whether this message was already acknowledged or not
- **AckRequired (Boolean)**: Indicates whether this message must be acknowledged or not
- **AckTime (Date)**: Date and time this alarm condition was acknowledged (or zero if it was not acknowledged)
- **ActiveSource (Integer)**: **-1 - None**, **0 - ActiveSource**, **1 - Scada**, **2 - Operator**, **3 - CCLink**, **4 - Billing**, **5 - Calculated**, **6 - Database**, **100 - TopologyProcessor**, **101 - PowerFlow**, **102 - StateEstimator**, **103 - LoadShedding**, **104 - DistLoadModelling**, **105 - SelfHealing**, or **106 - ExternalReader**
- **ActorID (String)**: User login that acknowledged this message (or an empty **String** if this message was still not acknowledged)
- **AlarmSourceName (String)**: Name of the Alarm Source object (only its name, not its full path)
- **Area (String)**: Area of this alarm
- **ChangeMask (Integer)**: Field not used by E3 currently, always in 0 (zero)
- **ConditionActive (Boolean)**: Indicates whether the alarm condition is active or not
- **ConditionName (String)**: Name of the last active alarm condition
- **Cookie (Integer)**: Identifies an Alarm Source during an execution session

- **CurrentValue (Double)**: Value of the Source at the moment this alarm condition became active
- **Enabled (Boolean)**: Indicates whether the alarm check on the Alarm Source is enabled or not
- **EventCategory (String)**: Name of the alarm category (for example, "Level", "Rate of Change", "Dead Band", "Digital", ou "Discrete")
- **EventTime (Date)**: Date and time of the last event update
- **EventTimeUTC (Date)**: Date and time of the last event update
- **EventType (String)**: "Event" or "Condition" (alarm)
- **FormattedValue (String)**: Contains the Source value (formatted) at the moment this alarm condition became active
- **FullAlarmSourceName (String)**: Full name of the Alarm Source object
- **InTime (Date)**: Date and time this alarm condition became active
- **Message (String)**: Alarm message
- **OutTime (Date)**: Date and time this condition left its alarm condition (or zero if it is still active)
- **Quality (String)**: "Good (xxx)", "Bad (xxx)", or "Uncertain (xxx)"
- **Severity (Integer)**: **0 - High**, **1 - Medium**, or **2 - Low**
- **Source (String)**: Link of the Alarm Source
- **SubConditionName (String)**: Name of this alarm's subcondition (for example, "LOLO", "LO", "HI", "HIHI", "DIG", etc.)

User-defined fields can also be used on a filter expression, by using the named defined on the Alarm Server.

Altogether, all messages that appear on an Alarm Filter list always pass through five filters:

- Filter by type (alarm or event) (the **FilterType** property)
- Filter by severity (the **ShowLowPriority**, **ShowMediumPriority**, and **ShowHighPriority** properties)
- Filter by area (the **AreaFilter** and **SimpleAreaFilter** properties)
- Filter by the **CustomFilter** property
- Filter by the alarm summary (equivalent to the expression "Enabled AND (ConditionActive OR (AckRequired AND NOT Acked))")

For usage examples of this property, please check E3Alarm's **CustomFilter** property.

5.6.2.2.14 Events

Returns a collection of Events on an Alarm Filter. For more information about the collection returned by this property, please check topic **Collection of Events**. Usage example of this property:

```
Sub CommandButton1_Click()
  For Each evt In Screen.Item("AlarmFilter1").Events
    str = ""
    str = str & "AlarmSourceName     = " &_
      evt.AlarmSourceName & Chr(13)
    str = str & "FullAlarmSourceName = " &_
      evt.FullAlarmSourceName & Chr(13)
    str = str & "EventTime           = " &_
      evt.EventTime & Chr(13)
    str = str & "EventTimeUTC        = " &_
      evt.EventTimeUTC & Chr(13)
    str = str & "InTime              = " &_
      evt.InTime & Chr(13)
    str = str & "OutTime             = " &_
      evt.OutTime & Chr(13)
    str = str & "AckTime             = " &_
      evt.AckTime & Chr(13)
    str = str & "CurrentValue        = " &_
      evt.CurrentValue & Chr(13)
    str = str & "ActorID             = " &_
      evt.ActorID & Chr(13)
    str = str & "Area                = " &_
      evt.Area & Chr(13)
    str = str & "ConditionName       = " &_
      evt.ConditionName & Chr(13)
    str = str & "EventCategory       = " &_
      evt.EventCategory & Chr(13)
    str = str & "EventType           = " &_
      evt.EventType & Chr(13)
    str = str & "Message             = " &_
      evt.Message & Chr(13)
    str = str & "Quality             = " &_
      evt.Quality & Chr(13)
    str = str & "Source              = " &_
      evt.Source & Chr(13)
    str = str & "SubConditionName    = " &_
      evt.SubConditionName & Chr(13)
    str = str & "FormattedValue      = " &_
      evt.FormattedValue & Chr(13)
    str = str & "UserField(1)        = " &_
      evt.UserField(1) & Chr(13)
    str = str & "UserField(2)        = " &_
      evt.UserField(2) & Chr(13)
```

```
    str = str & "UserField(3)        = " &_
      evt.UserField(3) & Chr(13)
    str = str & "UserField(4)        = " &_
      evt.UserField(4) & Chr(13)
    str = str & "Severity            = " &_
      evt.Severity & Chr(13)
    str = str & "Acked               = " &_
      evt.Acked & Chr(13)
    str = str & "AckRequired         = " &_
      evt.AckRequired & Chr(13)
    str = str & "ConditionActive     = " &_
      evt.ConditionActive & Chr(13)
    str = str & "Enabled             = " &_
      evt.Enabled & Chr(13)
    str = str & "EventCLSID          = " &_
      evt.EventCLSID & Chr(13)
    MsgBox str
  Next
End Sub
```

5.6.2.2.15 Filters

Returns a collection of Filters on an Alarm Filter. For more information about the collection returned by this property, please check topic **Collection of Filters**.

5.6.2.2.16 FilterType

Performs the alarm filters. Available options are the following:

- **1 - OnlyAlarms**: Displays only alarms
- **2 - OnlyEvents**: Displays only events
- **3 - AlarmsAndEvents**: Displays both alarms and events

5.6.2.2.17 FourthSortAscending

When this property is set to False, the sort order of events by a fourth field is in descending order. Otherwise, it is ascending. Default value of this property is False.

5.6.2.2.18 FourthSortField

Determines a fourth field for sorting events on an Alarm Filter. The name of this field must be always specified in English (please check all available fields on **E3 User's Manual**). Default value of this property is an empty **String**. This property has no effect when the **PrimarySortField**, **SecondarySortField**, or **ThirdSortField** properties are configured with an empty **String**.

5.6.2.2.19 InactiveHighNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **High** severity. This is a read-only property.

5.6.2.2.20 InactiveLowNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **Low** severity. This is a read-only property.

5.6.2.2.21 InactiveMedNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **Medium** severity. This is a read-only property.

5.6.2.2.22 InactiveNACKAlarms

9 Determines the total amount of inactive and unacknowledged alarms. This is a read-only property.

5.6.2.2.23 PrimarySortAscending

◩ When this property is set to False, the sort order of events by the primary field is in descending order. Otherwise, the sort order is in ascending order. Default value of this property is False.

5.6.2.2.24 PrimarySortField

A Determines a primary field for sorting events on an Alarm Filter. The name of this field must be always specified in English (please check all available fields on **E3 User's Manual**). Default value of this property is "EventTime". When this property is an empty **String**, the **SecondarySortField**, **ThirdSortField**, and **FourthSortField** properties have no effect.

5.6.2.2.25 SecondarySortAscending

◩ When this property is set to True, the sort order of events by a secondary field is in ascending order. Otherwise, it is descending. Default value of this property is False.

5.6.2.2.26 SecondarySortField

A Determines a secondary field for sorting events on an Alarm Filter. The name of this field must be always specified in English (please check all available fields on **E3 User's Manual**). Default value of this property is an empty **String**. This property has no effect when the **PrimarySortField** property is configured with an empty **String**.

5.6.2.2.27 ShowHighPriority

◩ Filters which alarms are displayed or not, according to their severity. When this property is set to True, **High**-severity alarms are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

5.6.2.2.28 ShowLowPriority

◩ Filters which alarms are displayed or not, according to their severity. When this property is set to True, **Low**-severity alarms are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

5.6.2.2.29 ShowMediumPriority

◩ Filters which alarms are displayed or not, according to their severity. When this property is set to True, **Medium**-severity alarms are displayed. Otherwise, these alarms are not displayed. Default value of this property is True.

5.6.2.2.30 SimpleAreaFilter

◩ When this property is set to True, the behavior of filtering by Alarm Area names is based only on the coincidence of the initial part of a name. When this property is set to False, this behavior considers the entire Area name, but allows using wildcard characters and multiple area filters, which must be separated by colons. Please check also the **AreaFilter** property, which specifies a filter by area name.

5.6.2.2.31 ThirdSortAscending

◩ When this property is set to False, the sort order of events by a third field is in descending order. Otherwise, it is ascending. Default value of this property is False.

5.6.2.2.32 ThirdSortField

A Determines a third field for sorting events on an Alarm Filter. The name of this field must be always specified in English (please check all available fields on **E3 User's Manual**). Default value of this property is an empty **String**. This property has no effect when the **PrimarySortField** or **SecondarySortField** properties are configured with an empty **String**.

5.6.3 Data Folder

The Data Folder object does not have specific events, methods, or properties, only general ones. These can be viewed on section **General Events, Methods, and Properties of Objects**.

5.6.4 Counter Tag

This section contains information about **events** and **properties** of the Counter Tag object. This object does not have methods associated to it.

5.6.4.1 Events

This section contains information about the events of the Counter Tag object.

5.6.4.1.1 OnPreset

OnPreset()

It occurs every time the **Preset** property value is reached.

5.6.4.2 Properties

This section contains information about the properties of the Counter Tag object.

5.6.4.2.1 AutoRestart

◩ Indicates that the counting must restart from zero after it reaches the value established in the **Preset** property. This property is only valid when the **CounterType** property is set to **0 - Preset**.

5.6.4.2.2 CounterType

Defines the counter's behavior. The available values for this property are:

- **0 - Preset**: the counting is stopped when the value set in the **Preset** property is reached
- **1 - Infinite**: the counting goes on indefinitely

5.6.4.2.3 Enabled

◩ Starts or stops the counter. If True, the counting is started, otherwise it is stopped.

5.6.4.2.4 Increment

9 Defines the update interval of the **Value** property. In case the value of this property is modified while the counting is on, this modification will only be effective when the counter is stopped and restarted.

5.6.4.2.5 Preset

9 Limit to be reached by the counter, in seconds. In case the **Preset** value is not a multiple of **Increment**, the Tag reaches this value before the next increment.

5.6.4.2.6 ResetCounterWhenEnabled

◩ Enables restarting the counting (from zero) each time the value of the **Enabled** property returns to True. When disabled, the counter restarts counting from the point it was previously interrupted.

5.6.4.2.7 Value

9 Displays the counting of the counter, in seconds. This is a read-only property. This property gets values multiples of **Increment**, except when the **Enabled** property is set to False. In this case, the value is the one when the Tag was disabled. When restarting the counting, the property value is the next multiple of **Increment**. The maximum value of this property is 2147483647 (0x7FFFFFFF).

5.6.5 Demo Tag

This section contains information about **methods** and **properties** of the Demo Tag object. This object does not have events associated to it.

5.6.5.1 Methods

This section contains information about the methods of the Demo Tag object.

5.6.5.1.1 Reset

Reset()

Resets the phase (movement in time) of Tag's wave format. This phase must not be reset unless the Tag is enabled. This method, when the Tag is enabled, has no effect on **CurrentTime**- and **Random**-type Tags, which are not periodic. When the Tag is disabled, its value is simply reset, regardless of Tag's type. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.Sine").Reset()
End Sub
```

5.6.5.2 Properties

This section contains information about the properties of the Demo Tag object.

5.6.5.2.1 Enabled

◩ Defines whether Demo Tag variation is activated. If True, a Demo Tag will update its **Value** property according to the settings in the **Period** and the **Scan** properties. Otherwise, variation is disabled. Default value is True. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Data.DemoTag1").Enabled = True
End Sub
```

5.6.5.2.2 Maximum

9 Determines the maximum Tag value. Default value is 100. Example:

```
Sub CommandButton2_Click()
  ' When clicking the button, a message box is opened,
  ' indicating DemoTag6 Maximum property value
  MsgBox Application.GetObject("Data.DemoTag6").Maximum
End Sub
```

5.6.5.2.3 Minimum

9 Determines the minimum Tag value. Default value is 0. Example:

```
Sub CommandButton2_Click()
  ' When clicking the button, a message box is opened,
  ' indicating DemoTag6 Minimum property value
  MsgBox Application.GetObject("Data.DemoTag6").Minimum
End Sub
```

5.6.5.2.4 Period

9 Defines the wave shape length, in milliseconds. Does not apply when the **Type** property is set to either **0 - Random** or **3 - CurrentTime**. Default value is 10,000 ms. Example:

```
Sub DemoTag1_OnStartRunning()
  Period = 1000
End Sub
```

5.6.5.2.5 Scan

9 Defines the time interval between two variations of the **Value** property, in milliseconds. Use this property to have a greater or smaller amount of data generated by the Demo Tag. Default value is 1,000. Scan value should be greater than 0. Example:

```
Sub Line1_Click()
  Application.GetObject("Data.DemoTag2").Scan = 200
End Sub
```

5.6.5.2.6 TimeStamp

The **TimeStamp** property is updated whenever a value or state change occurs in the **Value** or **Quality** properties. It informs what is the date and time associated to the Demo Tag value, as well as to its quality. This is a read-only property. Default value is 00:00:00.

5.6.5.2.7 Type

Determines the Tag's wave type. Change this property according to the next table. When the **Type** property is set to **3** (**CurrentTime**), the **Value** property will contain the current server's date and time.

Available options for Type

VALUE	WAVEFORM
0	Random
1	Sine
2	Square
3	CurrentTime
4	RampUp
5	RampDown
6	RampUpDown

Example:

```
Sub Line1_Click()
  Application.GetObject("Data.DemoTag2").Type = 2
End Sub
```

5.6.5.2.8 Value

The **Value** property varies according to the wave form type, established in the **Type** property. This is a read-only property. Default value is 0. Example:

```
Sub CommandButton1_Click()
  MsgBox Application.GetObject("Data.DemoTag2").Value = 10
End Sub
```

5.6.6 Internal Tag

This section contains information about **methods** and **properties** of the Internal Tag object. This object does not have events associated to it.

5.6.6.1 Methods

This section contains information about the methods of the Internal Tag object.

5.6.6.1.1 WriteEx

WriteEx(*NewValue*, *NewTimestamp*, *NewQuality*)

Allows changes to an Internal Tag's value, time stamp, and quality in a single operation. This method returns a **Boolean** indicating whether the operation has been successful or not.

The *NewValue* parameter specifies the Tag's new value; if omitted, Tag's value does not change. The *NewTimestamp* parameter specifies the Tag's new times tamp; if omitted, the time stamp from the moment of method's call is then used. The *NewQuality* parameter specifies Tag's new quality; if omitted, a Good quality (192) is then assumed. All these parameters can be omitted. Example:

```
Sub CommandButton12_Click()
  Dim Ret
  Ret = Application.GetObject("Data.InternalTag1")._
    WriteEx(123.456, "1/1/2001", 193)
  If Ret Then
    MsgBox "It works!"
  Else
    MsgBox "Failed!"
  End If
End Sub
```

5.6.6.2 Properties

This section contains information about the properties of the Internal Tag object.

5.6.6.2.1 Quality

9 Informs the value's quality in the **Value** property. This is a read-write property, but whenever the Internal Tag value is modified, by script or by a Link, it will be updated as well. Example:

```
Sub CommandButton1_Click()
  MsgBox Application.GetObject("Data.InternalTag1").Quality
End Sub
```

NOTE: For more information on quality, see the **Quality** topic on **E3 User's Manual**.

5.6.6.2.2 Retentive

◩ If True, the Internal Tag's value will be stored automatically, in case the active Domain stops. This guarantees that the Tag's value will be synchronized with the Standby server. Then, when the server starts running, Tag's value will be the same as the server that has stopped. Otherwise, Tag's value will be adjusted to the initial value whenever the Domain has been executed, or the active server switches. This property has no effect if changed at run time. Example:

```
Sub CommandButton1_Click()
  Dim status
  status = Application.GetObject("Data.InternalTag1").Retentive
  MsgBox status
  Select Case status
    Case True
      MsgBox "The internal tag value will be _
        stored automatically."
    Case False
      MsgBox "The tag value will be adjusted to the initial _
        value whenever the Domain is executed or _
        when an active server switch occurs."
  End Select
End Sub
```

NOTE: This property is only valid for Internal Tags contained on a Server. Internal Tags contained on a Viewer cannot be retentive.

5.6.6.2.3 TimeStamp

The **TimeStamp** property informs the date and time associated to the value in the **Value** property. This is a read-write property, but whenever an Internal Tag value is modified, by script or by a Link, it will be updated as well.

5.6.6.2.4 Value

This is a **Variant**-type property, which allows storing values of any type, from an **Integer** to an object pointer (see the next example). Use it to store values in the Viewer or in the Server, and to exchange data among several points in the application. Default value is empty. This is a read-write property. Example:

```
Sub Months_OnStartRunning()
  ' Months is an Internal tag.
  ' The event is used to initialize the array.
  Value = Array("January", "February", "March", _
    "April", "May", "June", "July", "August", _
    "September", "October", "November", "December")
End Sub
```

5.6.7 Timer Tag

This section contains information about **events** and **properties** of the Timer Tag object. This object does not have methods associated to it.

5.6.7.1 Events

This section contains information about the events of the Timer Tag object.

5.6.7.1.1 OnPreset

OnPreset()

It occurs every time the value of **NextExecTime** property is reached.

5.6.7.2 Properties

This section contains information about the properties of the Timer Tag object.

5.6.7.2.1 Enabled

Enables or disables the Timer Tag. The default value of this property is True.

5.6.7.2.2 NextExecTime

Displays the next triggering time. This is a read-only property.

5.6.7.2.3 RepeatInterval

This is used whenever the Tag type is set to **1 - ttContinuous**. The default value of this property is 00:00:01.

5.6.7.2.4 StartTime

Initial Timer Tag's date and time. For the **Single**-type, it is the triggering date and time itself. For other types, it is the moment when the Timer Tag triggers. The default value is the local date and time.

5.6.7.2.5 TriggerType

This is the Timer Tag's trigger type. The available options are:

Available options for TriggerType

OPTION	DESCRIPTION
0 - ttSingle	Single trigger.
1 - ttContinuous	Continuous trigger.
2 - ttDaily	Daily-based trigger.
3 - ttMonthly	Monthly-based trigger.

5.7 Database

This section contains information about **methods** and **properties** of the Database object. This object does not have events associated to it.

5.7.1 Methods

This section contains information about the methods of the Database object.

5.7.1.1 SetDBParameters

SetDBParameters(*ServerName*, *UserName*, *Password*, *DBName*)

A connection **String** with a Database server in the properties of a Database object. The *ServerName* parameter determines a server name. The *UserName* parameter determines a user name. The *Password* parameter determines a password to connect to this database. The *DBName* parameter is the name of a database used in a SQL Server. For other databases, this parameter is not used.

5.7.2 Properties

This section contains information about the properties of the Database object.

5.7.2.1 ConnectionActive

◩ Indicates whether E3 has an active connection with the database. E3 usually uses two connections with this database, one for writing and another one for reading. The **ConnectionActive** property is in True if at least one of those two connections is connected and working. This property should not be used to detect connection failures, as it may be in False in several situations, such as:

- When the database is not in use (no writing or reading operation was executed)
- When the database was just configured (when a connection property of the database changes at run time all connections are closed, and they are reconnected only at the next reading or writing operation)
- When the database connection dropped (for example, a network is not available or the database was closed)
- When the DBServer object is disabled (although a connection, even with a disabled DBServer object, can be reconnected if an application requests an immediate reading or writing to the database)

NOTE: If an application generates database operations consistently, that is, if it is always generating new writings or readings, the **ConnectionActive** property correctly reports the database's connection status, as this eliminates other situations when this property could be in False.

5.7.2.2 EnableSynchronization

Indicates to E3, if enabled (True), that it must also perform a data recording on a second database simultaneously, to improve security. If this property is enabled and there is a Standby server, E3 then performs a syncing between databases from both servers. The default value of this property is False, that is, syncing is disabled.

5.7.2.3 NetLibrary

Configures the type of network library of a Database. The available options are described on the next table.

Available options for NetLibrary property

OPTION	DESCRIPTION
0 - Default	Default library type
1 - NamedPipes	**Named Pipes**-type library
2 - TcpIp	**Winsock TCP/IP**-type library
3 - SpxIpx	**SPX/IPX**-type library
4 - BanyanVines	**Banyan Vines**-type library
5 - MultiProtocol	**Multi-protocolo (RPC)**-type library

NOTE: This **NetLibrary** property is only available on SQL Server databases.

5.7.2.4 nRetries

This property specifies the number of times E3 tries to perform a database operation, in addition to the first one. The default value of this property is 5 (five). If the value of this property is equal to 0 (zero), only one try is performed by operation. In case there is a database connection loss during any of these tries, that operation is aborted and the remaining attempts are dismissed.

5.7.2.5 ReconnectDelay

The **ReconnectDelay** property determines a delay time (in milliseconds) for an application to try to restore a lost database connection. The default value of this property is 2000 (two seconds).

5.7.2.6 SourceDatabase

For an **Access**-type database, this is the name of an .mdb file. For a **SQL Server**-type database, this is the name of a SQL Server, concatenated with the chosen database, such as Server/Database. For an **Oracle**-type database, this is the name of a connection. This is a read-only property.

5.7.2.7 SourceType

Indicates the database used by this object. The available options are the following:

- **0 - stAccess**: Access database
- **1 - stOracle**: Oracle database
- **2 - stSqlServer**: SQL Server database

5.7.2.8 TimeOutCommand

Contains a delay time for any database operation, before an application generates a *timeout* error. The default value is 180 (three minutes).

5.7.2.9 TimeOutConnection

Contains a delay time to perform a database connection, before an application generates a *timeout* error. The default value is 15 seconds.

5.7.2.10 TotalFailedWrites

Indicates the total amount of operations on .e3o files that failed and were discarded since the database connection was activated. This property can be automatically zeroed in several situations, such as:

- When deactivating the DBServer object
- If the E3DBEngine process is closed for any reason
- If configuration parameters of the DBServer object's connection were changed

5.7.2.11 UserName

A login used to connect to a Database. This is a read-only property.

5.7.2.12 UseTransaction

Defines whether a Database Server uses database transactions or not. If this property is set to True, each block of database operations (200 operations), such as Historic, Storage, Formula, and Alarm operations are executed at once, that is, as a single transaction.

5.8 Historic

This section contains information about **methods** and **properties** of the Historic object. This object does not have events associated to it.

5.8.1 Methods

This section contains information about the methods of the Historic object.

5.8.1.1 StartAcquisition

StartAcquisition()

Enables a Historic to store its field values periodically, using a rate specified in the **ScanTime** property. This method can be used at any time after using the **StopAcquisition** method. Default behavior of this method is to start an application enabled, that is, this method is always executed internally when starting a Historic. Example:

```
Sub Button1_Click()
  ' When clicking this button, enables Historic.
  Application.GetObject("Hist1").StartAcquisition()
End Sub
```

5.8.1.2 StopAcquisition

StopAcquisition()

Disables storing records by period in a Historic, regardless of the value specified in the **ScanTime** property. Storing by period is disabled until the **StartAcquisition** method is used. Default behavior of a Historic object is start an application with record storage disabled. Example:

```
Sub Hist1_OnStartRunning()
  ' Disables Historic as soon as it starts.
  StopAcquisition()
End Sub
```

5.8.1.3 WriteRecord

WriteRecord()

Inserts a new record on a Database. Values are obtained from current values of each variable specified as data sources for Historic fields. This method is used in two situations:

- To write a new data record before the time configured for the next recording when a Historic is enabled by time
- To write a new set of data when a Historic is disabled

Example:

```
Sub Tag1_OnValueChange()
  ' Writes a new record on a Historic
  ' when a Tag changes its value.
  Application.GetObject("Hist1").WriteRecord()
End Sub
```

5.8.2 Properties

This section contains information about the properties of the Historic object.

5.8.2.1 BackupDiscardInterval

Indicates a maximum time interval (minutes, hours, days, or months) for data on the backup table until it is discarded, regardless of the time data remains on the main table. For example, to keep data for 24 months on the main table and six more months on the backup table, this option's value must be 30 months. This property works with the **BackupDiscardTimeUnit** property. Default value of this property is 12 (twelve time units indicated by the **BackupDiscardTimeUnit** property).

NOTE: The total time indicated by combining the **BackupDiscardInterval** an **BackupDiscardTimeUnit** properties must be greater than the time indicated by the **DiscardInterval** and **DiscardTimeUnit** properties.

5.8.2.2 BackupDiscardTimeUnit

The **BackupDiscardTimeUnit** property indicates the time unit in which backup data remains stored until it is discarded. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

This property works with the **BackupDiscardInterval** property.

5.8.2.3 CacheSize

Defines the size of a record's block that must be read by a Historic before sending it to a database. For example, if **CacheSize** is equal to 4 (four), blocks with four records each are sent to the associated Database Server object. Valid values for this property must be in the range between 1 (one) and 4 (four). Default value of this property is 1 (one).

NOTE: A record block is sent every second, even if it does not reach the size configured in the **CacheSize** property.

5.8.2.4 CompressedTable

◩ Enables using dead band for data recording. Default value of this property is False.

5.8.2.5 DBServer

A Indicates a Database object used by this Historic to create tables and to store data. Default value of this property is an empty **String**.

5.8.2.6 DeadBand

9 This property works with the **CompressedTable** property. Indicates the calculated value over the last stored value (as a percentage) that defines whether this new value is stored. If the stored value is not numeric, changing it forces all values to be stored.

5.8.2.7 DiscardInterval

9 This property works with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates a time interval in which Historic data is stored on a database table, until it is discarded. Default value of this property is 1 (one time unit indicated by the **DiscardTimeUnit** property). If this property is set to a value less than the value of the **BackupDiscardInterval** property, E3 automatically sets the value of the **BackupDiscardInterval** property as double the value of the **DiscardInterval** property. This property can be modified at run time.

5.8.2.8 DiscardTimeUnit

This property works with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates a time unit in which table data remains stored until it is discarded. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

5.8.2.9 EnableBackupTable

◩ Creates a backup table where discarded data remains for security reasons. If set to True, this table is created. Otherwise, there is no backup table. Default value of

this property is False.

5.8.2.10 EnableDiscard

◩ Indicates Historic data discarding after a certain period of time. If set to False, data is stored indefinitely on a table. Otherwise, it is discarded after a certain period of time. Default value of this property is False.

5.8.2.11 EnableQualityLogs

◩ When in True, a Historic starts and E3 generates a record equal to the first collected record, but with bad quality (0) and a timestamp of minus one second.

5.8.2.12 ScanTime

Defines a time interval variation, in milliseconds, that a Historic waits to perform acquisition and storage of a new record on a table. Use this property if there is a need for more or less amount of data generated per second. Default value of this property is 1000.

5.8.2.13 TableName

A Defines a name for a table used in this Historic.

5.8.2.14 UserTable

◩ When this property is set to True, identifies this Historic as a user's Historic, that is, its table data was imported from a database. Otherwise, it is an E3's regular Historic. This is a read-only property.

5.8.2.15 UseTagQuality

◩ If this property is set to True, a Historic uses Tag source's quality value. Otherwise, the old evaluation method is used (**0: Uncertain value** or **1: Good value**).

5.8.2.16 VerificationInterval

9 This property works with the **VerificationUnit** property to control the time interval that E3 checks for older data, and then discard them. Default value of this property is 1 (one time unit indicated by the **VerificationUnit** property).

5.8.2.17 VerificationUnit

This property works with the **VerificationInterval** property. The **VerificationUnit** property indicates a time unit in which a data discard verification is performed. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

5.9 Storage

This section contains information about **methods** and **properties** of the Storage object. This object does not have events associated to it.

5.9.1 Methods

This section contains information about the methods of the Storage object.

5.9.1.1 CreateNewSession

CreateNewSession([*DefaultType*[, *DefaultMinRecTime*[, *DefaultMaxRecTime*[, *DefaultDeadBand*[, *DefaultDeadBandUnit*[, *DefaultScanTime*]]]]]])

Creates a **Session** capable of including data on a Storage, independent of normal acquisition. Optional parameters are used to configure Session Tags, if they are not informed during creation. They are, respectively:

- **DefaultType**: Data type (**0: Double**, **1: Bit**, **2: String**, or **3: Integer**). If no value is informed, then **0: Double** is used
- **DefaultMinRecTime**: Minimum time interval between recordings. If no value is informed, then 0 (zero) is used
- **DefaultMaxRecTime**: Maximum time interval without recordings. If no value is informed, then 3600 is used
- **DefaultDeadBand**: Dead band. If no value is informed, then 1 (one) is used
- **DefaultDeadBandUnit**: Tag's dead band unit (**0: Percentage** or **1: Absolute**). If no value is informed, then **1: Absolute** is used
- **DefaultScanTime**: Scan time. If no value is informed, then 0 (zero) is used

This methods works even if there are no fields configured in a Storage object.

5.9.1.2 StartAcquisition

StartAcquisition()

Starts or resumes data generation that is sent to a Database. A Storage receives notifications about which Tags were modified and, when this happens, it checks

whether these records were stored or not. When this method is used, change notifications and record generation are started or resumed. Example:

```
Sub Button1_Click()
  ' When clicking this button, enables Storage
  Application.GetObject("Storage1").StartAcquisition()
End Sub
```

5.9.1.3 StopAcquisition

StopAcquisition()

Stops data generation that is sent to a Database. A Storage receives notifications about which Tags were modified and, when this happens, it checks whether these records were recorded or not. When this method is used, change notifications and recording generation are stalled. Example:

```
Sub Storage1_OnStartRunning()
  ' Disables this Storage as soon as it starts
  StopAcquisition()
End Sub
```

5.9.2 Properties

This section contains information about the properties of the Storage object.

5.9.2.1 BackupDiscardInterval

9 Indicates a maximum time interval (minutes, hours, days, or months) for data on the backup table until it is discarded, regardless of the time data remains on the main table. For example, to keep data for 24 months on the main table and six more months on the backup table, this option's value must be 30 months. This property works with the **BackupDiscardTimeUnit** property. Default value of this property is 12 (twelve time units indicated by the **BackupDiscardTimeUnit** property).

NOTE: The total time indicated by the combination of **BackupDiscardInterval** and **BackupDiscardTimeUnit** properties must be greater than the time indicated by **DiscardInterval** and **DiscardTimeUnit** properties.

5.9.2.2 BackupDiscardTimeUnit

The **BackupDiscardTimeUnit** property indicates the time unit in which backup data remains stored until it is discarded. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

This property works with the **BackupDiscardInterval** property.

5.9.2.3 CacheSize

9 Defines the size of a record's block that must be read by a Storage before sending it to a database. For example, if **CacheSize** is equal to 4 (four), blocks with four records each are sent to the associated Database Server object. Default value of this property is 10.

NOTE: A record block is sent every second, even if it does not reach the size configured in the **CacheSize** property.

5.9.2.4 CompressionRate

9 Displays the data compression rate obtained until now.

5.9.2.5 DBServer

A Indicates a Database object used by this Storage to create tables and store data. Default value of this property is an empty **String**.

5.9.2.6 DiscardInterval

9 This property works with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates a time interval in which Historic data remains stored on a database table, until it is discarded. Default value of this property is 1 (one time unit indicated by the **DiscardTimeUnit** property). If this property is set to a value less or equal to the value of **BackupDiscardInterval** property, E3 automatically adjusts the value of the **BackupDiscardInterval** property as double the value of the **DiscardInterval** property. This property can be modified at run time.

5.9.2.7 DiscardTimeUnit

This property works with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates a time unit in which data remains stored until it is discarded. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

5.9.2.8 EnableBackupTable

Creates a backup table where discarded data remains for security reasons. If set to True, this table is created. Otherwise, there is no backup table. Default value of this property is False.

5.9.2.9 EnableDiscard

Enables Storage data discarding after a certain period of time. If set to False, data is stored indefinitely on a table. Otherwise, data is discarded after a certain period of time. Default value of this property is False.

5.9.2.10 Fields

Collection that points to Fields created in this Storage. For each Field, it is possible to view **Name** and **Link** properties, and modify **Type**, **MinRecTime**, **MaxRecTime**, **DeadBand**, **ScanTime**, and **DeadBandUnit** properties.

5.9.2.11 StringFieldSize

This property specifies the maximum size that Storage's **String**-type fields can have (this is the size used when creating a **Value** field of the **Strings** table).

5.9.2.12 TableName

Defines a name for a table used in this Storage.

5.9.2.13 VerificationInterval

This property works with the **VerificationUnit** property to control the time interval in which E3 checks how old data is, to discard it later. Default value of this property is 1 (one time unit indicated by the **VerificationUnit** property).

5.9.2.14 VerificationUnit

This property works with the **VerificationInternal** property. The **VerificationUnit** property indicates a time unit in which a data discard verification is performed. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

5.9.3 Storage Fields

This section contains information about **properties** of the Storage Field object. This object does not have events nor methods associated to it.

5.9.3.1 Properties

This section contains information about the properties of the Storage Field object.

5.9.3.1.1 DeadBand

9 Dead band used to calculate Storage's algorithm. Indicates the precision that users are willing to lose in each Field being stored. The higher the value of **DeadBand** property, the more compact the database (less data is recorded). This value can be specified in absolute units or as a percentage of Tag's current value, according to the **DeadBandUnit** property.

5.9.3.1.2 DeadBandUnit

9 Unit of **DeadBand** property. It can be specified in absolute units or as a percentage of Tag's current value.

5.9.3.1.3 Link

A Determines a data source associated to this Field. This is a read-only property.

5.9.3.1.4 MaxRecTime

9 Maximum difference, in seconds, between timestamps of two consecutive records stored on a database, that is, the maximum time with any data being recorded. For example, if a Tag value is not varying its value, yet its current value must be written to a database when reaching the number of seconds configured in **MaxRecTime**. This behavior can be disabled by using the value 0 (zero).

5.9.3.1.5 MinRecTime

9 Minimum difference, in milliseconds, between timestamps of any two records with the same quality stored on a database, that is, the minimum time for a new data to be stored. This parameter is used to limit the number of records written to a database, in case of a Tag undergoing sudden variations on its value. This behavior can be disabled by using the value 0 (zero).

5.9.3.1.6 Name

A Property that returns the name configured for this Field in a Storage object. By using this property, it is possible to search for items in the Collection of Storage

Fields (the **Fields** property).

5.9.3.1.7 ScanTime

9 Returns or sets Field's scan time in milliseconds, that is, how often a Tag's value is sent to the compression algorithm, in case it is not varying. If this property's value is equal to 0 (zero), the value of **MaxRecTime** property is used for the same purpose.

5.9.3.1.8 Type

Returns the type of object in the Storage format (**0: Double**, **1: Bit**, **2: String**, or **3: Integer**). This is a read-write property, but it only accepts changes while data collecting for this Field has not started yet.

NOTE: **Bit**, **String**, and **Integer** data types are not submitted to Storage's compression algorithm. Therefore, when there is a variation on Field's quality or value, this value is stored on a database. **DeadBand**, **DeadBandUnit**, **MaxRecTime**, and **MinRecTime** properties do not affect these data types, because they are exclusive to the compression algorithm.

5.9.4 Storage Session

This section contains information about **methods** of the Storage Session object. This object does not have events nor properties associated to it.

5.9.4.1 Methods

This section contains information about the methods of the Storage Session object.

5.9.4.1.1 AddField

AddField(*FieldName*[, *Type*, *MinRecTime*, *MaxRecTime*, *DeadBand*, *DeadBandUnit*, *ScanTime*])

This method is responsible for inserting temporary Tags on Session's structure. If its optional parameters are not informed, the values defined during the creation of this Session using Storage's **CreateNewSession** method are used. This method's parameters are described on the next table.

Parameters of the AddField method

PARAMETER	DESCRIPTION
FieldName	Name of the temporary Tag (mandatory).
Type	Tag type (optional). Possible values are **0: Double**, **1: Bit**, **2: String**, or **3: Integer**. Default value of this parameter is **0: Double**.

PARAMETER	DESCRIPTION
MinRecTime	Minimum time interval between recordings (optional). Default value of this parameter is 0 (zero).
MaxRecTime	Maximum time interval without recordings (optional). Default value of this parameter is 3600.
DeadBand	Temporary Tag's dead band (optional). Default value of this parameter is 1 (one).
DeadBandUnit	Temporary Tag's dead band unit (optional). Possible values are **0: Percentage** or **1: Absolute**. Default value of this parameter is **1: Absolute**.
ScanTime	Temporary Tag's scan time (optional). Default value of this parameter is 0 (zero).

This method returns True if a Tag was correctly added to this Session and False otherwise.

5.9.4.1.2 AddValue

AddValue(*FieldName*, *Timestamp*, *Quality*, *Value*)

Adds a value to a temporary Tag on this Session. This method's parameters are described on the next table.

Parameters of the AddValue method

PARAMETER	DESCRIPTION
FieldName	Name of a field to which this value is added. This name must exist on the original Storage's configuration, or else it must have been previously added using the **AddField** method.
Timestamp	Timestamp of a value to add.
Quality	Quality of a value to add.
Value	Value to add.

This method returns True if this value was correctly added and False otherwise.

5.9.4.1.3 Commit

Commit()

Stores all data kept in memory by this Session on Storage's database. This method

returns True if data was correctly stored and False otherwise.

5.10 Formulas

This section contains information about **methods** and **properties** of the Formula object. This object does not have events associated to it.

5.10.1 Methods

This section contains information about the methods of the Formula object.

5.10.1.1 CreateUnit

CreateUnit(*UnitName*)

Creates a unit on the Formula table. This method has the *UnitName* parameter which determines the name of the unit to be created. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, creates a new Unit
  Application.GetObject("Formula1").CreateUnit("Unit2")
End Sub
```

5.10.1.2 CreateValue

CreateValue(*ValueName*)

Creates a value set on the Formula table. This method has the *ValueName* parameter, which determines the name of the set to be created. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, creates a new Value
  Application.GetObject("Formula1").CreateValue("Template5")
End Sub
```

5.10.1.3 DeleteUnit

DeleteUnit(*UnitName*)

Deletes a unit on the Formula table. This method has the *UnitName* parameter, which informs the name of the unit to be deleted. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, deletes the unit
  Application.GetObject("Formula1").DeleteUnit("Unit2")
End Sub
```

5.10.1.4 DeleteValue

DeleteValue(*ValueName*)

Deletes a value set on the Formula table. This method has the *ValueName* parameter, which informs the value set to be deleted. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, deletes a value set
  Application.GetObject("Formula1").DeleteValue("Template5")
End Sub
```

5.10.1.5 FindUnit

FindUnit(*UnitName*)

Checks if a certain unit exists on the Formula database. This method has the *UnitName* parameter, which determines the name of the unit to be found. This method returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, displays a message box
  ' with the result
  MsgBox(Application.GetObject("Formula1")._
    FindUnit("Unit2"))
End Sub
```

5.10.1.6 FindValue

FindValue(*ValueName*)

Checks if a certain value set exists on Formula database. This method has the *ValueName* parameter, which informs the name of the set to be checked. Returns True if the operation is successful. Otherwise returns False. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button displays a message box
  ' with the result
  MsgBox CStr(Application.GetObject("Formula1")._
    FindValue("Template5"))
End Sub
```

5.10.1.7 GetUnitData

GetUnitData(*UnitName*, *TemplateName*, *Val*)

Places on the variable indicated by *Val* the Tag associated to the *UnitName* unit of

the *TemplateName* template. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim whatTag, whatFormula
  Application.GetObject("Formula1").GetUnitData _
    "Unit1", "Template2", var1
End Sub
```

5.10.1.8 GetValueData

GetValueData(*ValueName*, *TemplateName*, *Val*)

Places on the *Val* variable the value of the *ValueName* value set which is associated to the *TemplateName* template. Returns True if the operation is successful. Otherwise, returns False. Example:

```
Sub Button1_Click()
  Dim Value, whatFormula
  Application.GetObject("Formula1").GetValueData _
    "Value4", "Template2", var1
End Sub
```

5.10.1.9 LoadFormulaValuesQuiet

LoadFormulaValuesQuiet(*UnitName*, *ValueName*)

Loads a value set to a destination unit, without displaying a message. This method has the *UnitName* parameter, which determines the name of the unit and the *ValueName* parameter, which determines the name of the value set. Returns True if the operation is successful, and False on failure (which does not necessarily mean a script error). Example:

```
Sub Button1_Click()
  Application.GetObject("Formula1").LoadFormulaValuesQuiet _
    "Unit3", "Value1"
End Sub
```

NOTE: This method is also acessible through the **Viewer** object.

5.10.1.10 RenameUnit

RenameUnit(*UnitName*, *NewUnitName*)

Renames a certain existing unit on the Formula table. Returns True if the operation is successful, or False otherwise. This method has the *UnitName* parameter, which determines the name of the unit to be found, and the *NewUnitName* parameter, which informs the new unit name. Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking the button, renames the unit
  Application._
```

```
    GetObject("Formula1").RenameUnit "Unit2", "Unit3"
End Sub
```

5.10.1.11 RenameValue

RenameValue(*ValueName*, *NewValueName*)

Renames a certain set of values which exists in the Formula table. Returns True if the operation is successful or False, otherwise. This method has the *ValueName* parameter, which is the name of the set of values and the *NewValueName* parameter, which is the new name of the set of values. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").RenameValue "Template5",
"TemplateABC"
End Sub
```

5.10.1.12 SaveFormulaValues

SaveFormulaValues(*UnitName*, *ValueName*[, *IgnoreErrors*])

This method saves the current values of the Tags from an origin unit into a set of values in the Formula table. This method does not check limits, in case the template has an absolute type restriction. The *UnitName* parameter is the name of the origin unit and the *ValueName* parameter is the set of values which is saved. Returns True if the operation is successful. Otherwise, returns False. The *IgnoreErrors* parameter, when in True, forces all values to be saved, even if there are errors in the Formula links. However, its default value is False. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1") SaveFormulaValues "Unit1",
"Value1"
End Sub
```

5.10.1.13 SetUnitData

SetUnitData(*UnitName*, *TemplateName*, *Data*)

Loads the identified Tag to a certain template, on a certain unit, in the Formula table. Returns True if the operation is successful or False, otherwise. This method has the *UnitName* parameter, which is the name of the unit, the *TemplateName* parameter, which is the name of the Tag template, and the *Data* parameter, which is the name of the variable that contains the Tag name. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").SetUnitData "Unit2",
"Template5", 50
End Sub
```

5.10.1.14 SetValueData

SetValueData(*ValueName*, *TemplateName*, *Data*)

Changes the value referring to a template defined for a certain set of values. This method checks for limit values, returning True if the operation is successful or False, otherwise. This method has the *ValueName* parameter, which is the name of the set of values, the *TemplateName* parameter, which determines the template name, and the *Data* parameter, which is the name of the variable that holds the value. Example:

```
Sub CommandButton1_Click()
  Application.GetObject("Formula1").SetValueData "Unit2",
"Template1", 100
End Sub
```

NOTE: For more information about the **SetValueData** method, please check the articles **KB 112**, **KB 1246**, and **KB 4946** from the **Elipse Knowledgebase** website.

5.10.2 Properties

This section contains information about the properties of the Formula object.

5.10.2.1 DBServer

A Indicates the name of the database where Formula information is stored, that is, units and value sets. The default value of this property is an empty **String**.

5.10.2.2 ImmediateExecute

◩ When enabled, the Formula object writes its records directly to the Database, without passing through operation queues (.e3i and .e3o files). This makes database operations to be viewed faster.

5.10.2.3 TableName

A Indicates the name of the tables where Formula information is stored. The default value of this property is an empty **String**.

5.11 Alarms

This section contains information about events, methods, and properties of the Alarm Configuration, Alarm Area, Alarm Source, and Alarm Server objects.

5.11.1 Alarm Configuration

The Alarm Configuration object does not contain specific events, methods, or properties, only general ones. These can be viewed on section **General Events, Methods, and Properties of Objects**.

5.11.2 Alarm Area

This section contains information about **properties** of the Alarm Area object. This object does not have events nor methods associated to it.

5.11.2.1 Properties

This section contains information about the properties of the Alarm Area object.

5.11.2.1.1 ActiveAlarms

Determines the number of active alarms in an Area. This is a read-only property.

5.11.2.1.2 ActiveHighAlarms

Indicates the number of active alarms with **High** severity. This is a read-only property.

5.11.2.1.3 ActiveHighNACKAlarms

Indicates the number of non-acknowledged alarms with **High** severity. This is a read-only property.

5.11.2.1.4 ActiveLowAlarms

Indicates the number of active alarms with **Low** severity. This is a read-only property.

5.11.2.1.5 ActiveLowNACKAlarms

Indicates the number of non-acknowledged alarms with **Low** severity. This is a read-only property.

5.11.2.1.6 ActiveMedAlarms

Indicates the number of active alarms with **Medium** severity. This is a read-only property.

5.11.2.1.7 ActiveMedNACKAlarms

Indicates the number of non-acknowledged alarms with **Medium** severity. This is a read-only property.

5.11.2.1.8 ActiveNACKAlarms

Indicates the number of non-acknowledged alarms in an Area. This is a read-only property.

5.11.2.1.9 Alarm

Indicates that there are active alarms inside an Area. If this option is set to True, there is at least one active alarm inside an Area, and the **ActiveAlarms** property performs a reading from a server, then indicating the amount of active alarms. Otherwise, the **ActiveNACKAlarms** property performs a reading on non-acknowledged alarms. This is a read-only property.

5.11.2.1.10 AlarmVerify

Enables a check on all alarms inside an Area. After enabling that check (True), if the value of the **ActiveAlarms** property is greater than 0 (zero), the server then checks for active alarms, as well as non-acknowledged ones, listing these last ones in the **ActiveNACKAlarms** property. This property is useful to avoid a cascade effect on some applications, where the occurrence of an event triggers a large amount of related alarms.

5.11.2.1.11 InactiveHighNACKAlarms

Indicates the number of active and unacknowledged alarms with **High** severity. This is a read-only property.

5.11.2.1.12 InactiveLowNACKAlarms

Indicates the number of active and unacknowledged alarms with **Low** severity. This is a read-only property.

5.11.2.1.13 InactiveMedNACKAlarms

Indicates the number of active and unacknowledged alarms with **Medium** severity. This is a read-only property.

5.11.2.1.14 InactiveNACKAlarms

Determines the total amount of active and unacknowledged alarms. This is a read-only property.

5.11.2.1.15 UserFields

Returns an object that is a collection of Alarm's User Fields of an Alarm Area. Please check topic **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.11.3 Alarm Source

This section contains information about common **methods** and **properties** of the Alarm Source object. This object does not have common events associated to it.

NOTE: When an Alarm Source object is disabled or deactivated, the **ActiveNACKAlarm**, **Alarm**, **CurrentSeverity**, **CurrentSubConditionName**, **FormattedValue**, **RawAlarm**, and **Value** properties assume their default values, indicating that the Alarm Source is not linked to an active alarm message. In this case, the values of the **Alarm** and **ActiveNACKAlarm** are propagated to the counters of upper areas.
For these properties to receive a value other than the default, it is necessary that the following conditions are present:

- The Alarm Source must be enabled and active
- The **Alarm Area** (and all its hierarchically superior objects) must be enabled and active
- The **Alarm Server** must be active

5.11.3.1 Common Methods

This section contains information about the common methods of the Alarm Source objects.

5.11.3.1.1 Ack

Ack([*ActorID*])

Acknowledges an alarm configured in an Alarm Source object. This method returns a **Boolean** indicating whether that operation was successful or not. The *ActorID* parameter informs the name of the user responsible for acknowledging the alarm. This is an optional parameter and, if omitted, assumes Viewer's user logon, "Anonymous" if there is no user logged on, or "System" if this method's call started at the server.

5.11.3.1.2 GetAlarm

GetAlarm()

Returns an object that grants access to every type of alarm's specific settings. This allows checking or modifying any alarm properties at run time. Depending on the type of alarm, this method returns the following properties:

- **Analog Alarm**: Responsible for digital alarm settings

Digital Alarm Properties

ITEM	DESCRIPTION
DigitalReturnMessageText	Returning message of a digital alarm
Digital	Enables or disables a check on a digital alarm
DigitalLimit	Limit for a digital alarm
DigitalMessageText	Text of a digital alarm's message
DigitalSeverity	Digital alarm's severity. Set of values: • **0**: High • **1**: Medium • **2**: Low
DigitalAckRequired	Acknowledgment required for this type of alarm (digital)

- **Analog Alarm:** Responsible for analog alarm settings. Properties of this object (it contains four alarm levels):

Analog Alarm properties

ITEM	DESCRIPTION
LevelDeadBand	Dead band for a level alarm
LevelReturnMessageText	Returning message of this alarm

LoLo Alarm (Very Low)

ITEM	DESCRIPTION
LoLo	Enables or disables a check on a **Very Low** alarm
LoLoLimit	Limit for a **Very Low** alarm
LoLoMessageText	Text of a **Very Low** alarm's message
LoLoSeverity	Importance of a **Very Low** alarm's level. Set of values: • **0**: High • **1**: Medium • **2**: Low
LoLoAckRequired	Acknowledgment required for this alarm level (**Very Low**)

Lo Alarm (Low)

ITEM	DESCRIPTION
Lo	Enables or disables a check on a **Low** alarm
LoLimit	Limit for a **Low** alarm
LoMessageText	Text of a **Low** alarm's message
LoSeverity	Importance of a **Low** alarm's level. Set of values: • **0**: High • **1**: Medium • **2**: Low
LoAckRequired	Acknowledgment required for this alarm level (**Low**)

Hi Alarm (High)

ITEM	DESCRIPTION
Hi	Enables or disables a check on a **High** alarm
HiLimit	Limit for a **High** alarm
HiMessageText	Text of a **High** alarm's message
HiSeverity	Importance of a **High** alarm's level. Set of values: • **0**: High • **1**: Medium • **2**: Low
HiAckRequired	Acknowledgment required for this alarm level (**High**)

HiHi Alarm (Very High)

ITEM	DESCRIPTION
HiHi	Enables or disables a check on a **Very High** alarm
HiHiLimit	Limit for a **Very High** alarm
HiHiMessageText	Text of a **Very High** alarm's message
HiHiSeverity	Importance of a **Very High** alarm's level. Set of values: • **0**: High • **1**: Medium • **2**: Low

ITEM	DESCRIPTION
HiHiAckRequired	Acknowledgment required for this alarm level (**Very High**)

- **Rate of Change Alarm**: Responsible for rate of change alarm settings

Rate of Change Alarm properties

ITEM	DESCRIPTION
ROCReturnMessageText	Returning message of a rate of change alarm
ROC	Enables or disables a check on a rate of change alarm
ROCLimit	Limit for a rate of change alarm. For this alarm to occur, it is enough that the value of the associated Tag exceeds this value in one second
ROCMessageText	Text of a rate of change alarm's message
ROCSeverity	Importance of a rate of change alarm. Set of values: • **0**: High • **1**: Medium • **2**: Low
ROCAckRequired	Acknowledgment required for this type of alarm (rate of change)

- **Dead Band Alarm**: Responsible for dead band alarm settings

Dead Band Alarm Properties

ITEM	DESCRIPTION
DeadBandSetPoint	Limit for a dead band alarm. Every time the value of the associated Tag exceeds the value of this property up or down the **DeadBandLimit** value, this alarm occurs
DeadBandReturnMessageText	Returning message of a dead band alarm
DeadBand	Enables or disables a check on a dead band alarm
DeadBandLimit	Limit for a dead band alarm
DeadBandMessageText	Text of a dead band alarm's message
DeadBandSeverity	Importance of a dead band alarm. Set of values: • **0**: High • **1**: Medium • **2**: Low

ITEM	DESCRIPTION
◩ **DeadBandAckRequired**	Acknowledgment required for this type of alarm (dead band)

Example:

```
Sub Button1_Click()
  Dim val
  ' When clicking this button, changes the Lo alarm level
  ' of BatteryLevel AlarmSource
  Application.GetObject("AlarmConfig1.Area1.BatteryLevel")_
    .GetAlarm().LoLimit = 10.2
End Sub
```

NOTE: Properties relative to each type of alarm can be accessed directly via scripts and Links, as well as viewed on object's Property List, thus it is not mandatory editing these properties via **GetAlarm** method anymore.

5.11.3.2 Common Properties

This section contains information about the common properties of the Alarm Source objects.

5.11.3.2.1 ActiveNACKAlarm

◩ If set to True, indicates that the Source was not acknowledged since its last activation. This is a read-only property. Default value of this property is False.

5.11.3.2.2 Alarm

◩ If set to True, indicates this Alarm's active condition. Default value of this property is False.

5.11.3.2.3 AlarmVerify

◩ If set to True, enables a check on this Alarm source, that is, generates the alarm.

5.11.3.2.4 AreaNameOverride

A Determines an alternative name for the Area that contains the Alarm Source. Default value of this property is an empty **String**.

NOTES:

- When this property is empty, the name of the Alarm Source is composed by the names of Area objects hierarchically above it.
- This property, even when filled in, does not influence Alarm Area counters hierarchically above it, which keep counting all alarms of this Alarm Source.
- If this property is changed at run time, its new value is only effective when the next alarm on this Alarm Source occurs.

5.11.3.2.5 CurrentSeverity

Indicates the last severity of the active alarm, that is:

- **0**: High
- **1**: Medium
- **2**: Low

Default value of this property is -1 (minus one), indicating that the Alarm Source is not active.

5.11.3.2.6 CurrentSubConditionName

A Determines the name of the last active alarm condition. Available options for this property are described on the next table.

Available options for CurrentSubConditionName

OPTION	DESCRIPTION
LOLO	Analog alarm on a **Very Low** range
LO	Analog alarm on a **Low** range
HI	Analog alarm on a **High** range
HIHI	Analog alarm on a **Very High** range
DB	Dead Band alarm
ROC	Rate Of Change alarm
DIG	Digital alarm

Default value of this property is an empty **String**.

NOTE: For **Discrete**-type Alarm Sources, this property assumes the name of the user-defined Subcondition (the **Discrete** tab of properties of this type of Alarm).

5.11.3.2.7 Delay

9 Specifies a delay time for the alarm, in ms, when entering and when leaving a condition. When this property is set to 0 (zero, which is always its default value), no

delay applies to it. When it is different from 0 (zero), the alarm is only activated or deactivated if it remains on the same condition for a time greater than or equal to the one specified.

5.11.3.2.8 DoubleAckRequired

◩ When set to True, indicates that this alarm can be acknowledged when active and when inactive, that is, can be doubly acknowledged. When set to False, indicates that this alarm can only be acknowledged once and, when acknowledged, it leaves the alarm's list. Alarms that do not need acknowledgment (by using the **AckRequired** property) does not allow that type of customization. Applications earlier than version 2.5 have this property set to False.

5.11.3.2.9 Event

◩ When set to True, indicates that this is an **Event**-type alarm. An **Event**-type alarm, when activated, is acknowledged by the "System" user. Therefore, when it is acknowledged nothing happens (it has no effect), as well as it does not increment the number of active alarms nor the number of unacknowledged alarms. This property cannot be modified at run time.

5.11.3.2.10 Format

A The **Format** property specifies the type of format applied to this object. It allows changing the way data is displayed without changing data behind it. This property can be manually edited or configured using the formatting window. Its usage is similar to formats used on spreadsheets, following the same basic syntax. Data types described on the next table are supported.

Data types supported by the Format property

DATA	DESCRIPTION
Numerical	Decimal, scientific, hexadecimal, binary, and octal output
Text	General text
Boolean	**Boolean** values
Date and Time	Gregorian calendar

5.11.3.2.11 FormattedValue

A Contains the value of the alarm as its **Value** property, formatted according to the **Format** property. This is a read-only property. Default value of this property is **Null**.

5.11.3.2.12 RawAlarm

◩ Indicates whether an alarm must be active, regardless of its delay. When this delay is equal to 0 (zero), **RawAlarm**'s value is the same as the **Alarm** property. This

is a read-only property. Default value of this property is False.

5.11.3.2.13 Source

A Contains an expression that must be evaluated to determine whether this alarm should occur or not.

5.11.3.2.14 UserFields

Returns an object that is a collection of Alarm's User Fields of an Alarm Source. Please check topic **Collection of Alarm's User Fields** for more information about the collection of objects returned by this property.

5.11.3.2.15 Value

Contains the value that was evaluated to determine whether this alarm occurs or not. Default value of this property is **Null**.

5.11.3.3 Analog Alarm Source

This section contains information about **properties** of the Analog Alarm Source object. This object does not have events nor methods associated to it.

When the value of the **Event** property (common to all Alarm Sources) is set to True, the **HiEvent**, **HiHiEvent**, **LoEvent**, and **LoLoEvent** properties cannot be modified (all alarm subconditions behave as events).

5.11.3.3.1 Properties

This section contains information about the properties of the Analog Alarm Source object.

5.11.3.3.1.1 Hi

Enables or disables a check on **Hi**-type alarms.

5.11.3.3.1.2 HiAckRequired

Indicates whether a **Hi**-type alarm requires acknowledgment or not.

5.11.3.3.1.3 HiEvent

Defines whether an Alarm's **Hi** subcondition must be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.4 HiHi

◩ Enables or disables a check on **HiHi**-type alarms.

5.11.3.3.1.5 HiHiAckRequired

◩ Indicates whether a **HiHi**-type alarm requires acknowledgment or not.

5.11.3.3.1.6 HiHiEvent

◩ Defines whether an Alarm's **HiHi** subcondition must be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains as True. In addition, this property cannot be modified at run time.

5.11.3.3.1.7 HiHiLimit

9 Indicates the level in which a **HiHi**-type alarm is activated.

5.11.3.3.1.8 HiHiMessageText

A Sets a text message for a **HiHi**-type alarm's limit.

5.11.3.3.1.9 HiHiSeverity

Indicates the level of importance of a **HiHi**-type alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.10 HiLimit

9 Indicates the level in which a **Hi**-type alarm is activated.

5.11.3.3.1.11 HiMessageText

A Sets a text message for a **Hi**-type alarm's limit.

5.11.3.3.1.12 HiSeverity

Indicates the level of importance of a **Hi**-type alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.13 LevelDeadBand

9 Dead band for alarm level limits.

5.11.3.3.1.14 LevelReturnMessageText

A Sets a returning message for alarm's level.

5.11.3.3.1.15 Lo

◩ Enables or disables a check on **Lo**-type alarms.

5.11.3.3.1.16 LoAckRequired

◩ Indicates whether a **Lo**-type alarm requires acknowledgment or not.

5.11.3.3.1.17 LoEvent

◩ Defines whether an Alarm's **Lo** subcondition must be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.18 LoLimit

9 Indicates the level in which a **Lo**-type alarm is activated.

5.11.3.3.1.19 LoLo

◩ Enables or disables a check on **LoLo**-type alarms.

5.11.3.3.1.20 LoLoAckRequired

◩ Indicates whether a **LoLo**-type alarm requires acknowledgment or not.

5.11.3.3.1.21 LoLoEvent

◩ Defines whether an Alarm's **LoLo** subcondition must be treated as an event. If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in True. In addition, this property cannot be modified at run time.

5.11.3.3.1.22 LoLoLimit

9 Indicates the level in which a **LoLo**-type alarm is activated.

5.11.3.3.1.23 LoLoMessageText

A Sets a text message for a **LoLo**-type alarm's limit.

5.11.3.3.1.24 LoLoSeverity

Indicates the level of importance of a **LoLo**-type alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.3.1.25 LoMessageText

A Sets a text message for a **Lo**-type alarm's limit.

5.11.3.3.1.26 LoSeverity

Indicates the level of importance of a **Lo**-type alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.4 Digital Alarm Source

This section contains information about **properties** of the Digital Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.4.1 Properties

This section contains information about the properties of the Digital Alarm Source object.

5.11.3.4.1.1 Digital

◩ Enables or disables a check on digital alarms.

5.11.3.4.1.2 DigitalAckRequired

◩ Indicates whether this digital alarm requires acknowledgment or not.

5.11.3.4.1.3 DigitalLimit

9 Limit for this digital alarm.

5.11.3.4.1.4 DigitalMessageText

A A text message for this digital alarm.

5.11.3.4.1.5 DigitalReturnMessageText

A A returning message for this digital alarm.

5.11.3.4.1.6 DigitalSeverity

Severity of this Digital Alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.5 Dead Band Alarm Source

This section contains information about **properties** of the Dead Band Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.5.1 Properties

This section contains information about the properties of the Dead Band Alarm Source object.

5.11.3.5.1.1 DeadBand

◩ Enables or disables a check on dead band alarms.

5.11.3.5.1.2 DeadBandAckRequired

◩ Indicates whether this dead band alarm requires acknowledgment or not.

5.11.3.5.1.3 DeadBandLimit

9 Limit for this dead band alarm.

5.11.3.5.1.4 DeadBandMessageText

A A text message for this dead band alarm.

5.11.3.5.1.5 DeadBandReturnMessageText

A A returning message for this dead band alarm.

5.11.3.5.1.6 DeadBandSetpoint

A Alarm's dead band limit. Each time the value of the associated Tag exceeds this property's value up or down the **DeadBandLimit** value, this alarm occurs.

5.11.3.5.1.7 DeadBandSeverity

9 Importance of this dead band alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.6 Rate Of Change Alarm Source

This section contains information about **properties** of the Rate Of Change Alarm Source object. This object does not have events nor methods associated to it.

5.11.3.6.1 Properties

This section contains information about the properties of the Rate Of Change Alarm Source object.

5.11.3.6.1.1 ROC

◩ Enables or disables a check on rate-of-change alarms.

5.11.3.6.1.2 ROCAckRequired

◩ Indicates whether this rate-of-change alarm requires acknowledgment or not.

5.11.3.6.1.3 ROCLimit

9 Limit for this rate-of-change alarm. For this alarm to occur, it is enough that the value of the associated Tag exceeds this value by one second.

5.11.3.6.1.4 ROCMessageText

A A text message for this rate-of-change alarm.

5.11.3.6.1.5 ROCReturnMessageText

A Returning message for this rate-of-change alarm.

5.11.3.6.1.6 ROCSeverity

Importance of this rate-of-change alarm. Available options for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.3.7 Discrete Alarm Source

This section contains information about **properties** of the Discrete Alarm Source object. This object does not have events nor methods associated to it.

When the value of the **Event** property (common to all Alarm Sources) is set to True, the **Kind** property of objects belonging to the collection of Subconditions of a Discrete Alarm Source cannot be modified (all alarm subconditions behave as events).

5.11.3.7.1 Properties

This section contains information about the properties of the Discrete Alarm Source object.

5.11.3.7.1.1 DiscreteReturnMessageText

A Returns or modifies the returning message of this Discrete Alarm Source. If any object in the collection of **Subconditions** has its **Kind** property set to **2** (two, **Return**), the **Message** property of that object is then used instead of the value defined in **DiscreteReturnMessageText**.

5.11.3.7.1.2 SubConditions

Returns an object that is a collection of Subconditions of this Discrete Alarm Source. Check topic **Collection of Subconditions** for more information about the collection of objects returned by this property.

5.11.3.7.2 Collection of Subconditions

This section contains information about **methods** and **properties** common to the collection of Subconditions returned by the **SubConditions** property of a Discrete Alarm Source.

5.11.3.7.2.1 Common Methods

This section contains information about the methods common to the collection of Subconditions of a Discrete Alarm Source.

AddSubCondition

AddSubCondition([*Name*, *Caption*, *Message*, *Kind*, *AckRequired*, *Severity*, *Value*])

Adds a Subcondition object to the collection of Subconditions. This method has the optional parameters described on the next table.

Parameters of the AddSubCondition method

PARAMETER	DESCRIPTION
Name	Object's name. Corresponds to Subcondition's **Name** property. If it is omitted, the Subcondition is then created as "Subcondition". If the value passed on this parameter already exists in the collection, it is automatically incremented.
Caption	Object's description. Corresponds to Subcondition's **Caption** property.
Message	Subcondition's message text. Corresponds to Subcondition's **Message** property.
Kind	The type of behavior of this Subcondition. Possible values for this parameter are **0: Alarm** (default), **1: Event**, or **2: Return**. Corresponds to Subcondition's **Kind** property.
AckRequired	Indicates whether this Subcondition requires acknowledgment. Corresponds to Subcondition's **AckRequired** property. Default value of this property is True.

PARAMETER	DESCRIPTION
Severity	The type of severity of this Subcondition. Possible values for this parameter are **0: High**, **1: Medium** (default), or **2: Low**. Corresponds to Subcondition's **Severity** property.
Value	Parameter containing a value that is evaluated to determine whether this alarm occurs or not. Corresponds to the **Value** property, common to all Alarm Sources.

Item

Item(*Index*)

Returns a reference to a Subcondition object, indicated by *Index*. This parameter can be an index in the collection (starting at 1), or the object's name (the **Name** property).

RemoveSubCondition

RemoveSubCondition(*Index*)

Removes the Subcondition object indicated by *Index*. This parameter can be an index in the collection (starting at 1), or the object's name (the **Name** property).

5.11.3.7.2.2 Common Properties

This section contains information about the properties of the collection of Subconditions of a Discrete Alarm Source.

Count

9 Returns the number of children objects (items) of a collection of Subconditions. This property works together with the **Item** method. If the collection has no children objects, returned value is 0 (zero).

5.11.3.7.2.3 SubCondition

This section contains information about **properties** of **Subcondition**-type objects inside the collection returned by the **SubConditions** property of a Discrete Alarm Source. This object does not have events nor methods associated to it.

Properties

This section contains information about the properties of the Subcondition object.

AckRequired

◩ Indicates whether this Subcondition object requires acknowledgment or not.

Caption

A Subcondition's description.

Enabled

◩ Enables or disables this Subcondition.

Kind

Indicates this Subcondition's behavior. Possible values for this property are the following:

- **0**: Alarm
- **1**: Event
- **2**: Return

If the **Event** property, common to all Alarm Sources, is set to True, this property cannot be modified, and its value always remains in **1** (one, **Event**). In addition, this property cannot be modified at run time.

Limit

9 Defines the Alarm Source's value to generate this Subcondition.

Message

A The event message when this Subcondition is active. If the **Kind** property is set to **2** (two, **Return**), this property is considered as the alarm's returning message.

Name

A Name of this Subcondition. This value is case-insensitive.

Severity

9 The type of severity of this Subcondition. Possible values for this property are the following:

- **0**: High
- **1**: Medium
- **2**: Low

5.11.4 Alarm Server

This section contains information about **methods** and **properties** of the Alarm Server object. This object does not events associated to it.

5.11.4.1 Methods

This section contains information about the methods of the Alarm Server object.

5.11.4.1.1 AckAllAlarms

AckAllAlarms([*ActorID*])

Performs an acknowledgment of all server alarms, regardless of their Area. This method returns a **Boolean** indicating whether this operation was successful or not. The *ActorID* parameter informs the name of the user responsible for acknowledging these alarms. This is an optional parameter and, if omitted, assumes Viewer's logged-on user, "Anonymous" if there is no user logged on, or "System" if this method's call started at the server. Example:

```
Sub Button1_Click()
  ' When clicking this button, acknowledges all alarms
  Application.GetObject("AlarmServer1")._
    AckAllAlarms (Application.User)
End Sub
```

5.11.4.1.2 AckArea

AckArea(*Area*[, *ActorID*])

Performs an acknowledgment of alarms on a certain Area. This method returns a **Boolean** indicating whether this operation was successful or not. The *Area* parameter specifies the name of the Area or Areas whose alarms are acknowledged, by comparing the beginning of their names. For example, **AckArea("ANA")** acknowledges alarms in Areas "ANALOG", "ANA.AREA2", etc. If this parameter is empty, this method behaves as **AckAllAlarms**. The *ActorID* parameter informs the name of the user responsible for acknowledging these alarms. This is an optional parameter and, if omitted, assumes Viewer's logged-on user, "Anonymous" if there is no user logged on, or "System" if this method's call started at the server. Example:

```
Sub Button1_Click()
  'When clicking this button, acknowledges Area1 alarms
  Application.GetObject("AlarmServer1")._
    AckArea "Area1", Application.User
End Sub
```

5.11.4.1.3 LogTrackingEvent

LogTrackingEvent(*Message*[, *ActorID*], *Area*, *Severity*, *EventTime*, *Source*, *EventCategory*, *EventType*, *UserFields*, *AlarmSourceName*, *FullAlarmSourceName*)

Simulates an event or an alarm and sends it straight to Alarm Server's database, without passing through an E3Alarm. Therefore, this event cannot be seen on an E3Alarm, nor can this alarm be acknowledged.

Each parameter of this method allows specifying the value of the field with the same

name in the event. Event fields are fulfilled according to this method's parameters, described on the next table.

Parameters of the LogTrackingEvent method

NAME	DESCRIPTION
Message	Text parameter that specifies the content of event's **Message** field. If omitted, assumes an empty **String**.
ActorID	Optional text parameter that specifies the content of event's **Operator** field. If omitted, assumes the user logged on Viewer, "Anonymous" if there is no user logged on, or "System" if this method's call started at the server.
Area	Text parameter that specifies the content of event's **Area** field. If omitted, assumes an empty **String**.
Severity	Numeric parameter that determines event's severity. If omitted, assumes the value 0 (zero), that is, high severity.
EventTime	Specifies an event's timestamp. If omitted, assumes the value of the timestamp at the moment this method was used.
Source	Text parameter that specifies the content of event's **Source** field. If omitted, assumes an empty **String**.
EventCategory	Text parameter that specifies an event's category. If omitted, assumes an empty **String**.
EventType	Text parameter that specifies an event's type. If omitted, it assumes the "Tracking" value.
UserFields	**Array**-type parameter, where each position assumes the value of a user-specified field.
AlarmSourceName	Text parameter that specifies an Alarm Source's name. If omitted, assumes an empty **String**.
FullAlarmSourceName	Text parameter that specifies an Alarm Source's full path, including Area, Alarm Configuration, and possible Folders where this Alarm is inserted. If omitted, assumes an empty **String**.

The other event fields cannot be specified and always assume the following values:

- **CurrentValue**: 0.0

- **Quality**: ""
- **ConditionActive**: 0 (False)
- **ConditionName**: ""
- **SubConditionName**: ""
- **Acked**: 1 (True)
- **AckRequired**: 0 (False)
- **Enabled**: 1 (True)
- **EventTimeUTC**: *Always equal to **EventTime** (such as in alarm events)
- **ChangeMask**: 0 (zero)
- **Cookie**: 0 (zero)

NOTE: This method fails if Alarm Server's **Store alarms on a database** option is not selected, or when storing data on a database fails.

Example:

```
Sub CommandButton1_Click()
  ' In UserFields parameter, for each array element,
  ' displays the value set to it
  Application.GetObject("AlarmServer1").LogTrackingEvent_
    "Button clicked", Application.User, "Operation", 2, ,_
    "Button1", , , array(1, 2, "a", "b")
End Sub
```

5.11.4.2 Properties

This section contains information about the properties of the Alarm Server object.

5.11.4.2.1 ActiveAlarms

9 Determines the total amount of active alarms in an application. This is a read-only property.

5.11.4.2.2 ActiveHighAlarms

9 Indicates the number of active alarms with **High** severity. This is a read-only property.

5.11.4.2.3 ActiveHighNACKAlarms

Indicates the number of unacknowledged alarms with **High** severity. This is a read-only property.

5.11.4.2.4 ActiveLowAlarms

Indicates the number of active alarms with **Low** severity. This is a read-only property.

5.11.4.2.5 ActiveLowNACKAlarms

Indicates the number of unacknowledged alarms with **Low** severity. This is a read-only property.

5.11.4.2.6 ActiveMedAlarms

Indicates the number of active alarms with **Medium** severity. This is a read-only property.

5.11.4.2.7 ActiveMedNACKAlarms

Indicates the number of unacknowledged alarms with **Medium** severity. This is a read-only property.

5.11.4.2.8 ActiveNACKAlarms

Indicates the total amount of unacknowledged alarms in an application (active or not). This is a read-only property.

5.11.4.2.9 BackupDiscardInterval

Indicates a maximum time interval (minutes, hours, days, or months) for data on the backup table until it is discarded, regardless of the time data remains on the main table. For example, to keep data for 24 months on the main table and six more months on the backup table, this option's value must be 30 months. This property works with the **BackupDiscardTimeUnit** property. Default value is 12 (twelve time units indicated by the **BackupDiscardTimeUnit** property).

NOTE: The total time indicated by the combination of **BackupDiscardInterval** and **BackupDiscardTimeUnit** properties must be longer than the time indicated by **DiscardInterval** e **DiscardTimeUnit** properties.

5.11.4.2.10 BackupDiscardTimeUnit

The **BackupDiscardTimeUnit** indicates a time unit in which backup data remains stored until they are discarded. The available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

This property works with the **BackupDiscardInterval** property.

5.11.4.2.11 DataSource

Defines a Database object to use for storing alarm data. Default value of this property is an empty **String**, that is, there is no Database to store data.

5.11.4.2.12 DiscardInterval

This property works with the **DiscardTimeUnit** property. The **DiscardInterval** property indicates a time interval in which alarm data is stored on a database table, until it is discarded. Default value of this property is 1 (one time unit indicated by the **DiscardTimeUnit** property). If this property is configured with a value less than or equal to the **BackupDiscardInterval** property, E3 automatically sets the value of **BackupDiscardInterval** as doubles the value of **DiscardInterval**. This property can be changed at run time.

5.11.4.2.13 DiscardTimeUnit

This property works with the **DiscardInterval** property. The **DiscardTimeUnit** property indicates a time unit in which table data remains stored until it is discarded. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

5.11.4.2.14 EnableBackupTable

Creates a backup table where discarded data is stored for security reasons. If set to True, this table is created. Otherwise, there is no backup table. Default value of this property is True.

5.11.4.2.15 EnableDiscard

◩ Indicates alarm's data discard after a certain time. If set to False, data is stored indefinitely on a table. Otherwise, it is discarded after a certain time. Default value of this property is False.

5.11.4.2.16 InactiveHighNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **High** severity. This is a read-only property.

5.11.4.2.17 InactiveLowNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **Low** severity. This is a read-only property.

5.11.4.2.18 InactiveMedNACKAlarms

9 Indicates the number of inactive and unacknowledged alarms with **Medium** severity. This is a read-only property.

5.11.4.2.19 InactiveNACKAlarms

9 Determines the total amount of inactive and unacknowledged alarms. This is a read-only property.

5.11.4.2.20 Logging

◩ Creates a record with information about alarms on a Database specified by the **DataSource** property. If set to False, this record is not created. Otherwise, this record is created. Default value of this property is False.

5.11.4.2.21 TableName

A Defines a name for an alarms table. Default value is "Alarms". It can be changed at run time and it is applied immediately.

5.11.4.2.22 VerificationInterval

9 This property works with the **VerificationUnit** property to control the time interval that E3 checks if data is old, and then discard it. Default value of this property is 1 (one time unit indicated by the **VerificationUnit** property).

5.11.4.2.23 VerificationUnit

This property works with the **VerificationInterval** property. The **VerificationUnit** property indicates a time unit to perform a verification to discard data. Available options are:

- **0 - dtHour**: Hours
- **1 - dtDay**: Days
- **2 - dtMonth**: Months (default)
- **3 - dtMinute**: Minutes

CHAPTER 6

Frequently Asked Questions

How to open a windowed Screen that displays a title bar with minimize, maximize, and close buttons?

To do so, users must use Splitter's **SetFrameOptions** method. The *Flags* parameter specifies features for this window. A value of 127 defines a window with visible **Minimize**, **Maximize**, and **Close** buttons.

How to open a modal Screen?

To open a modal Screen, use Viewer's **DoModal** method. For example:

```
Application.DoModal "Screen1", "Title1", 0, 0, 400, 200, 0, 1
```

This code opens a Screen named "Screen1", with title "Title1", at position (0, 0), with a width of 400 pixels and a height of 200 pixels, passes 0 (zero) as a Screen parameter, and enables a window's title bar.

How to copy values from an E3Browser's row to a Tag?

First, select the row (or record) on an E3Browser. Then, use E3Browser's **GetColumnValue** method. The *Index* parameter is the column's index to copy (starting at zero).

How to prevent users from typing a String in a SetPoint?

Check if the typed value is a number on SetPoint's **Validate** event. For example:

```
Sub Text1_Validate(Cancel, NewValue)
  If NOT IsNumeric(newValue) Then
    MsgBox "This value must be a number."
    Cancel = True
  End If
End Sub
```

How to open a date picker to select a date and a time when clicking a SetPoint?

By using the **ShowDatePicker** method on SetPoint's **Click** event. For example:

```
Sub Text1_Click()
  Dim datevalue
  If Application.ShowDatePicker(datevalue) Then
    Value = datevalue
  End If
End Sub
```

How to acknowledge all alarms of an Area?

To acknowledge all alarms of an Area by script, users can use the **AckArea** method. This method has two parameters, as follows:

- *Area* is the name of the Alarm Area that that users want to acknowledge an alarm
- *User* is the name of the logged-on operator, which can be the **Application.User** item

To acknowledge all active alarms, users can also use the **AckAllAlarms** method.

How to execute an action when clicking a specific mouse button or a key?

By using Screen's **KeyDown** or **KeyUp** events. These events are triggered when a key is pressed or released and return two parameters. One is the code of the pressed key, and the other one is the SHIFT and CTRL key status when this key was pressed. The idea is to compare event's returned parameters with the code of the expected character.

How to create a WhileRunning script?

By creating an event linked to a property that always has the same value. For example, the **Visible** property of a Screen object. While this object is visible (its **Visible** property set to True), this script is executed. However, it is recommended to avoid using **WhileRunning** scripts, because they may affect application's performance. In most cases, they can be replaced by Links.

How to create an OnValueChanged script?

By creating an event linked to a Tag's **Value** property, which is executed when this property changes its value.

NOTE: Be careful to not use Viewer methods on Server, such as the **MsgBox** method. If this is the case, this event can be created on Screen or even on Viewer object, instead of creating it on a Tag.

How to create Tags and Screen objects at run time?

By using the **AddObject** method. For example, the following script creates I/O Tags on **Driver1** Driver.

```
Set obj = Application.GetObject("Driver1")
For i = 1 To 100
  Set tag = obj.AddObject("IOTag", false)
  tag.Name = "IOTag" & CStr(i)
```

```
  tag.Activate
Next
```

How to display a message on Screen when changing a Tag's value?

By creating an event on the Screen linked to Tag's **Value** property, which is executed when this property changes its value. On this event, use the **MsgBox** method to display that message.

How to create a Query with a filter by date before assembling a Report?

To do so, users must configure the Query object (please check the **Query** chapter) that accompanies a Report and then create the necessary variables on the **Filter** column. On the event that opens this Report, use a script similar to this one:

```
Set report = Application.LoadReport("[Report1]")
Set query = report.Query()
query.SetVariableValue "Variable1", Value1
query.SetVariableValue "Variable2", Value2
report.PrintPreview()
```

Where:

- *[Report1]* is the name of the Report to open
- *Variable1* and *Variable2* are variables created on **E3TimeStamp** field's filter
- *Value1* and *Value2* are dates to query

To check other types of filters, please check the **Query** chapter or the available documentation at **Elipse Knowledgebase**.

How to debug errors on Server and Viewer scripts?

If an event is executing on Viewer, use the **MsgBox** method. If an event is executing on server, use the **Trace** method.

Headquarters
Rua 24 de Outubro, 353 - 10º andar
90510-002 Porto Alegre
Phone: (+55 51) 3346-4699
Fax: (+55 51) 3222-6226
E-mail: elipse-rs@elipse.com.br

Taiwan
9F., No.12, Beiping 2nd St., Sanmin Dist.
807 Kaohsiung City - Taiwan
Phone: (+886 7) 323-8468
Fax: (+886 7) 323-9656
E-mail: evan@elipse.com.br

Check our website for information about a representative in your city or country.

www.elipse.com.br

kb.elipse.com.br

forum.elipse.com.br

www.youtube.com/elipsesoftware

elipse@elipse.com.br

Gartner 2014 Cool Vendor

OPC FOUNDATION MEMBER

Gartner, Cool Vendors in Brazil 2014, April 2014.
Gartner does not endorse any vendor, product or service depicted in its research publications, and does not advise technology users to select only those vendors with the highest ratings. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact. Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability of fitness for a particular purpose.

Microsoft Partner
Gold Independent Software Vendor (ISV)